# Packer Classification based on Association Rule Mining

Khanh Huu The Dam[a], Thomas Given-Wilson[a], Axel Legay[a], Rosana Veroneze[a,b,*]

[a]*Louvain School of Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium*
[b]*School of Electrical and Computer Engineering, University of Campinas, Campinas, SP, Brazil*

**Abstract**

Detecting packer programs is a key step in the defense against malicious programs and plays a key role in cyber security defenses. One challenge with packer classification is that many features have been used and their individual significance is unknown. An effective approach for building classifiers without requiring prior knowledge of feature significance is to use associative classification (AC) algorithms, which combine association rules and classification. This work considers many different AC algorithms for the challenge of packer detection. Novel variations of many of these algorithms are also developed to address challenges related to having many features of varying types. The effectiveness of the classifiers produced by these algorithms is evaluated, including over time as packers and malware evolve.

*Keywords:* Malware, packer detection, pattern mining, association rules, associative classification

## 1. Introduction

Detecting packer programs is a key step in the defense against malicious programs and plays a key role in cyber security defenses. Many malicious programs use packing to hide their malicious behavior and intent from analysis and to evade detection [38, 39]. Thus being able to detect packed programs and identify the packer is a key step in identifying threats and potentially malicious programs [11, 13].

One challenge with packer classification is that there are many features that have been used and their individual significance is unknown [11, 12, 13, 17, 24, 26, 28, 36, 42, 44, 46, 50, 51, 57]. This makes the construction of effective detection and classification algorithms complex and unclear. In particular many works approach the challenge with an a priori assumption about which features may be useful in detection or classification. Some recent work has explored up to 119 different features and many different machine learning approaches [13], but is still limited to only considering course grained feature selection.

An effective approach for building classifiers from heterogeneous high-dimensional data without requiring prior knowledge of feature significance is to use association rules. This work considers

*Corresponding author
*Email addresses:* khan.dam@uclouvain.be (Khanh Huu The Dam), thomas.given-wilson@uclouvain.be (Thomas Given-Wilson), axel.legay@uclouvain.be (Axel Legay), veroneze@dca.fee.unicamp.br (Rosana Veroneze)

different *associative classification* (AC) techniques [2, 33, 40] to build effective multi-class classifiers for the challenge of packer detection. The AC techniques combine association rule mining [35] and classification to yield highly interpretable and accurate classifiers [2, 40]. Essentially, the output of an AC algorithm is a set of if-then rules, which are easy to understand and interpret. The rules, called *class association rules* (CARs), represent the correlations among different attributes and the class attribute simultaneously. Further, this approach also provides a confidence probability for each rule that can be used in the classification [15]. For these reasons, AC algorithms usually produce higher accuracy than decision trees and other rule-based classifiers [2, 15, 48]. Moreover, generally these have the advantage that they do not need to consider properties of the features, and can build rules without detailed expert knowledge and assumptions about the relative significance or value of features.

In contrast to global approaches, such as feature selection or data space transformations, AC proposals offer a way of selecting all (and only) relevant subspaces to address the curse of high-dimensionality [22]. Thus, AC algorithms decrease the over-fitting risk by not including irrelevant subspaces and decrease the under-fitting risk by selecting all relevant subspaces [22].

There are a variety of different AC algorithms that take different approaches to rule generation and rule selection (classifier building) [40]. This work explores three well-known AC algorithms, CBA [33], CMAR [30] and PAM [15], for packer classification. Both CBA and CMAR prioritize more general rules, while PAM prioritizes more specific rules. CBA and PAM tend to select fewer rules for the classifier, while CMAR tends to select more rules. All of them use a heuristic based on sample coverage in the classifier building stage.

Novel variations of these AC algorithms are also developed to address challenges related to packer classification, such as having many features, mixed data types, and very unbalanced data sets. First of all, we explore a novel pattern mining algorithm, called RIn-Close_CVC [54, 55], in the rule generation stage of PAM. RIn-Close_CVC can **directly** handle data sets with attributes characterized by distinct distributions or even mixed data types, i.e., without transforming the original relation between the rows and columns of a data set into binary relations. The numerical attributes are handled with online partitioning instead of a priori partitioning, which is commonly used in traditional pattern mining algorithms [35]. We also propose variations related to the classifier builders of these proposals. One of these variations is a simplification of the PAM's classifier builder stage. The other is the addition of a rule to the classifiers built by CMAR, which assigns a default class label for samples that do not match any rule of the classifier. Lastly, we propose a novel algorithm, called *best rule* (BR), that adopts a simple and efficient strategy to build the classifier using the metric *principality* proposed in [15].

The effectiveness of the classifiers produced by these AC algorithms is evaluated on a large data set of $\sim 240,000$ samples. The effectiveness is considered when exploiting different parameters in

rule generation, and over eight different data sets each of approximately $30,000$ samples. The overall accuracy, and weighted class accuracy is considered, to explore which algorithms are better in practice, and which are better when the data set labels are less representative. Further, the effectiveness is considered over time, exploring how the classifiers lose effectiveness as the packers and samples evolve over time.

The main contributions of the paper are as follows.

- Four new variations of existing CMAR and PAM-based AC algorithms.
- The novel *best rule* (BR) algorithm.
- The evaluation of these three original and six new algorithms on eight data sets of representative samples of packer malicious software.
- Demonstrating that BR (and variation) are highly effective on representative data sets, improving over existing algorithms.

The structure of the paper is as follows. Section 2 recalls common background useful for understanding this work. Section 3 presents the related works. Section 4 presents the proposed methods. Section 4.1 presents the best rule algorithm (BR). Section 4.2 introduces new variations to algorithms including CBA, CMAR, PAM and BR. Section 5 overviews the data sets used in this work. Section 6 details the evaluation methodology. Section 7 presents the results. Section 8 discusses the results and lessons learned. Section 9 concludes and discusses future work. Appendix A presents didactic examples of all main concepts used in this work.

## 2. Common Background

Liu et al. [33] connected association rule mining (ARM) and classification to form a field known as *associative classification* (AC). The unseen instances are classified based on *association rules* (ARs) whose consequent parts describe the class attribute, named *class association rules* (CARs). Usually, in order to obtain this kind of classifiers, the complete set of CARs should be first discovered from the training data set and then a subset of such rules is selected to form the classifier [53]. The first stage is called *rule generator* and the second one is called *classifier builder*.

Let $\mathcal{I} = \{x_1, x_2, \cdots, x_m\}$ be a set of elements called items. A set $X \subseteq \mathcal{I}$ is called an *itemset*. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ be another set of elements called transaction identifiers (*tids*). A set $T \subseteq \mathcal{T}$ is called a *tidset*. Let $\mathbf{D}$ be a binary relation on the set of tids and items, i.e., $\mathbf{D} \subseteq \mathcal{T} \times \mathcal{I}$. Thus, $(t, x) \in \mathbf{D}$ if the transaction $t$ contains the item $x$.

Traditional pattern mining algorithms handle only binary relations on the set of tids and items. Thus, when dealing with a non-binary data set, each *item* is usually given by an attribute-value pair $(A_i, v)$, where $A_i$ is an attribute taking a value $v$. For a discrete attribute, all the possible values will give rise to an item. For a continuous attribute, its value range is first partitioned and, subsequently, all the possible values will give rise to an item.

Let $\mathbf{i} : 2^{\mathcal{T}} \rightarrow 2^{\mathcal{I}}$ be a function defined as follows:

$$\mathbf{i}(T) = \{x | \forall t \in T, (t, x) \in \mathbf{D}\}, \tag{1}$$

i.e., $\mathbf{i}(T)$ is the set of items that are common to all the transactions in the tidset $T$. Let $\mathbf{t} : 2^{\mathcal{I}} \rightarrow 2^{\mathcal{T}}$ be a function defined as follows:

$$\mathbf{t}(X) = \{t | \forall x \in X, (t, x) \in \mathbf{D}\}, \tag{2}$$

i.e., $\mathbf{t}(X)$ is the set of tids that contain all the items in the itemset $X$.

The support of an itemset $X$, denoted $sup(X)$, is the number of transactions that contain $X$, i.e., $sup(X) = |\mathbf{t}(X)|$.

An itemset $X$ is said to be *frequent* if $sup(X) \geq minsup$, where $minsup$ is a user-defined threshold. The problem of mining all frequent itemsets can be described as follows: determine all subsets $X \subseteq \mathcal{I}$ such that $sup(X) \geq minsup$.

Sets $X \subseteq \mathcal{I}$ and $T \subseteq \mathcal{T}$, such that $X = \mathbf{i}(\mathbf{t}(X))$ and $T = \mathbf{t}(\mathbf{i}(T))$, are said to be closed. Thus, an itemset $X$ is a *closed itemset* if $X = \mathbf{i}(\mathbf{t}(X))$.

Let $X$ be a closed itemset, then all subsets $Y \subseteq X$ such that $\mathbf{t}(Y) = \mathbf{t}(X)$ are called generators of $X$. The set of all generators of $X$ is called the *equivalence class* of $X$, denoted by $Equiv(X) = \{Y | Y \subseteq X \text{ and } \mathbf{t}(Y) = \mathbf{t}(X)\}$. An itemset $Y \in Equiv(X)$ is called a *minimal generator* (MG) or *key* of $X$ if $\nexists Z \in Equiv(X)$ such that $Z \subset Y$. In an equivalence class there can be several minimal generators, but only one closed itemset, which is the maximal element in this equivalence class. An equivalence class is called trivial if it consists only of a closed itemset [37].

**Proposition 1.** *Each itemset $Y \subseteq \mathcal{I}$ belongs to only one equivalence class, which is given by $Y \in Equiv(\mathbf{i}(\mathbf{t}(Y)))$.*

The (worst-case) complexity of pattern mining algorithms, such as Apriori [6], depends on the number of frequent itemsets in the input data set. The worst-case scenario is when the number of frequent itemsets is $2^{|\mathcal{I}|} - 1$. However, the number of frequent itemsets is much smaller than that in practice. Also, all equivalence classes are trivial in the worst-case scenario, so the number of closed itemsets would be equal to the number of itemsets. However, in practice, the set of all closed frequent

itemsets (CFIs) can be orders of magnitude smaller than the set of all frequent itemsets [58].

For mining closed itemsets, we have algorithms with a polynomial complexity of $O(|\mathcal{T}||\mathcal{I}|^2)$ [29] (or in $O(|\mathcal{T}|^2|\mathcal{I}|)$). This means that the total time complexity of the algorithm is $O(\mathfrak{c}|\mathcal{T}||\mathcal{I}|^2)$, where $\mathfrak{c}$ is the number of closed itemsets in the input data set. This is the case, for instance, of the well-known LCM [52, 25] and In-Close5 [8] algorithms.

An *association rule* (AR) is an expression of the form $X \to Y$, where $X$ and $Y$ are itemsets and $X \cap Y = \emptyset$. $X$ is called antecedent and $Y$ is called consequent of the rule. The support of an association rule $X \to Y$ is the number of transactions that contain the itemset $X \cup Y$: $sup(X \to Y) = sup(X \cup Y)$. The confidence of an association rule $X \to Y$ measures its predictive accuracy and is given by $conf(X \to Y) = sup(X \cup Y)/sup(X)$. A rule is considered a strong rule if $conf(X \to Y) \geq minconf$, where $minconf$ is a user-defined threshold. The completeness (or recall) of an association rule $X \to Y$ denotes the percentage of data records containing $Y$ that contains $X$ and $Y$ at the same time, and is is given by $comp(X \to Y) = sup(X \cup Y)/sup(Y)$.

It is sometimes desirable to extract co-occurrences between a set of items and a specific target attribute of interest. These rules are known as *class association rules* (CARs) and are expressed in the form $X \to (y, c)$, where $X$ is an itemset, $y$ is the target attribute (class attribute) and $c$ is a class label. For convenience, we write $X \to c$. For CARs, the antecedent part of the rule is also called *condition*.

Let $X$ be a closed itemset. For all CARs $Y \to c$ with $Y \in Equiv(X)$: $sup(Y \to c) = sup(X \to c)$, $conf(Y \to c) = conf(X \to c)$ and $comp(Y \to c) = comp(X \to c)$ since $\mathbf{t}(Y) = \mathbf{t}(X)$. Among these CARs, the one with the closed itemset in the antecedent part of the rule is the most **specific** one, and those with a minimal generator in the antecedent part of the rule are the most **general** ones.

## 3. Related Work

This section considers some related works. These are divided into two areas, the first is on the challenge of multi-class packer classification. The second is on the algorithms and associative classifiers, with an emphasis on the algorithms that will be used as competitors of our proposals.

### *3.1. Multi-Class Packer Classification*

Entropy based approaches to detecting (but not classifying) packers started with Lyda and Hamrock [36]. Successive works also used entropy to detect packed or unpacked samples [17, 26, 44, 50, 51]. Later Hubballi and Dogra [24] extended this to packer classification as is done here with their accuracy close to 96% on a data set of $\sim$ 5500 samples.

Perdisci et al. [42] and Zakeri et al. [57] applied machine learning techniques on a variety of features for packer detection (again not classification). Sun et al. [46] considered the challenge of packer

classification and since then other works have since also considered classification [12, 28]. More recently Bat-Erdene et al. [11] also considered if multiple packers had been used on the same program. The best results of these works achieved $\sim 98\%$ accuracy on a data set of $\sim 700$ samples.

Biondi et al. [13] observed that many features had been considered and so experimented on all 119 features in the literature; the same features considered in this work. Biondi et al.'s approach was to consider many machine learning techniques on different subsets of features and a data set of around $280,000$ samples collected in the same manner as those used in this work. Their F-scores for the best classifiers comparable to the results here range from 0.968 to 0.997 and with similar drop off in effectiveness on later data than the training period. Regarding this drop off in accuracy, this work considers five months whereas Biondi et al. only consider three, with this work indicating a more significant decrease after the first three months (it is unknown if the same holds for Biondi et al. since they did not consider a longer time frame). One main challenge Biondi et al. identify is the selection of features and the cost of picking the right ones, something addressed here naturally by the rule generation that explicitly states the features of significance.

### *3.2. Associative Classification Algorithms*

There are many different AC proposals, and most of them share a two-step process: rule generator and classifier builder stages [40].

In the first stage, rules are obtained using ARM. Remark that a CAR is a special case of an AR, in which only the class attribute is considered in the consequent part of the rule. As the problem to be solved is simpler, some adaptations to the ARM algorithms are usually proposed. These adaptations are commonly related to some rules' precision threshold, such as minimum confidence. Thus, in the rule generator stage, many proposals share the same idea with small differences [40]. For instance, CBA and PAM use an Apriori-like [6] algorithm, while CMAR uses a FP-Growth-like algorithm [21]. But both, Apriori and FP-Growth, have the same goal: to mine all frequent itemsets. In any case, the complexity of generating the rules depends upon the pattern mining algorithm, as explained in Section 2.

In the second stage, there are many more differences among the AC proposals since each one uses a different strategy to perform the post-processing of the rules obtained in the first stage [40]. Alwidian et al. [7] clustered these differences in the following aspects: rule ranking, rule pruning and prediction method.

Three usual factors in the rule ranking are: confidence, support and generality/specificity of the rule. As far as we know, most of the AC algorithms prioritizes general rules. Some examples are CBA, CMAR, CBA2 [34], WCBA [7], MCAR [47] and FACA [19]. However PAM and ACCF [31] are

examples that prioritize specific rules. In $L^3$ [10], the selection between general and specific rules is a tuning option, with the specific rules being the default option.

It is known that the most general rules are those built on minimal generators [43], while the most specific rules are those built on closed itemsets. But, interestingly, this information is not well explored by the AC proposals. An exception is ACCF [31] that uses a Charm-like algorithm (Charm [58] is a pattern mining algorithm that enumerates all **closed** frequent itemsets).

The data set coverage is adopted in the rule pruning of many AC proposals, such as CBA, CMAR, PAM, CBA2, WCBA, MCAR and FACA. Diversely, $L^3$ performs a lazy pruning, where only rules that exclusively misclassify training data are discarded.

There are also several prediction methods used in AC algorithms. With their specifics, some AC methods employ the highest ranked rule in the classifier and others use multiple rules [2], which is called *rule lists* and *rule sets*, respectively [9]. Some examples of the first group are CBA, PAM, CBA2, MCAR and ACCF, while some examples of the second group are CMAR, WCBA and FACA.

Although not the majority of the AC proposals, we have some proposals that are oriented to a specific domain application. Some examples where AC algorithms were successfully applied are: malware detection [56], text classification [4], phishing websites detection [3, 5, 19], and breast cancer prediction [7]. In [56], where AC was used for malware detection, the analyzed data set represents Windows portable executable files, having two class labels: benign or malicious executables. The authors extracted the API (Application Program Interface) calls as the features of the file samples.

#### *3.2.1. Classification Based on Associations (CBA)*

For the rule generator stage, *Classification based on associations* (CBA) [33] uses a modified version of Apriori [6], in which each itemset is evaluated with an associated target value (the class attribute). In this stage, minimum support ($minsup$) and confidence ($minconf$) thresholds are considered, so the mined CARs are both frequent and strong. For all the rules that have the same condition, only the rule with the highest confidence is chosen. For instance: let $R_1 : \{x_1, x_2\} \rightarrow c_1$, $R_2 : \{x_1, x_2\} \rightarrow c_2$, $conf(R_1) = 55\%$ and $conf(R_2) = 45\%$, then only $R_1$ would be selected to the next stage. If there are more than one rule with the same highest confidence, CBA randomly selects one rule.

For the classifier builder, first of all, CBA ranks the rules as follows. Given two rules $R_i$ and $R_j$, $R_i$ is prior to $R_j$ if:

1. $conf(R_i) > conf(R_j)$, or
2. $conf(R_i) = conf(R_j)$ and $sup(R_i) > sup(R_j)$, or
3. $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$ and $R_i$ is generated earlier than $R_j$.

The time complexity to rank the rules depends on the sorting algorithm. For typical serial sorting algorithms, a good worst-case complexity would be $O(\kappa \log \kappa)$, where $\kappa$ is the number of rules mined from the training data set.

Each rule is analyzed according to its ranking. If a rule can correctly classify at least one data record, it will be added to the classifier. Those data records that are covered by the selected rule are removed from the training data set. Finally, the procedure ends by discarding those rules that do not improve the final accuracy of the resulting classifier. Thus, the time complexity to build the classifier is $O(\kappa|\mathcal{T}||\mathcal{I}|)$.

**Lemma 1.** *The condition of a CAR selected to the CBA's classifier is a minimal generator of minimum size.*

*Proof.* Remark that each itemset $Y \subseteq \mathcal{I}$ belongs to only one equivalence class. Let $X$ be a closed itemset, only one itemset $Y \in Equiv(X)$ can be the condition of a selected CAR $Y \rightarrow c$ since the classifier builder removes all data records that are covered by the rule and a selected rule must correctly classify at least one data record. Suppose $Y$ is not a minimal generator of minimum size. It contradicts the rule rank since among the rules with the same confidence and support, it selects the one generated earlier by an Apriori-based algorithm. □

After the selection of the rules, the classifier has the form: $\langle R_1, R_2, \cdots, default\_class \rangle$, where $default\_class$ is the default class (usually the dominant class label in the training data set). In classifying an unseen case, the first rule that satisfies the case will classify it. If no rules are matching the unseen case, the default class is assigned. Thus, CBA provides a simple rule list where a new instance is predicted in $O(\eta|\mathcal{I}|)$, where $\eta$ is the number of rules in the classifier.

### *3.2.2. Classification Based on Multiple Association Rules (CMAR)*

*Classification based on Multiple Association Rules* (CMAR) [30] extends the FP-Growth [21] for the rule generator stage in such a way that the mined CARs are also both frequent and strong.

For the classifier builder, CMAR ranks the rules as follows. Given two rules $R_i$ and $R_j$, $R_i$ is prior to $R_j$ if:

1. $conf(R_i) > conf(R_j)$, or
2. $conf(R_i) = conf(R_j)$ and $sup(R_i) > sup(R_j)$, or
3. $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$ and $len(R_i) < len(R_j)$,

where $len(R)$ denotes the number of items in rule $R$.

After the rule ranking, CMAR employs the following two methods for rule pruning. First, given two rules $R_i$ and $R_j$, where $R_i$ is a general rule w.r.t. $R_j$, CMAR prunes $R_j$ if $R_i$ has higher rank than

$R_j$ and $conf(R_i) > conf(R_j)$. Second, for each rule $R : X \rightarrow c$, CMAR tests whether $X$ is positively correlated with $c$ by $\chi^2$ testing. Those rules with $\chi^2$ not passing a significance level threshold $\gamma$ are pruned. The CR-Tree, a prefix tree structure used by CMAR, speeds up these two pruning methods.

CMAR also uses a heuristic based on the data set coverage to select the rules for the classifier. The data set coverage heuristic used in CMAR is similar to the one used in CBA. The major difference is that, instead of removing one data object from the training data set immediately after it is covered by some selected rule, CMAR lets it stay there until it is covered by at least $\delta$ rules. Let $\kappa_*$ be the number of rules after the rule pruning, so the complexity of this heuristic would be $O(\delta\kappa_*|\mathcal{T}||\mathcal{I}|)$.

To classify a new data object, CMAR collects the subset of rules matching the new object from the set of selected rules for classification. Trivially, if all the rules matching the new object have the same class label, CMAR just simply assigns that label to the new object. If the rules are not consistent in their class labels, CMAR divides the rules into groups according to class labels. CMAR compares the effects of the groups and yields to the strongest group. CMAR uses a weighted $\chi^2$ metric to measure the strength of each group. Thus, CMAR provides a rule set whose prediction of a new instance is slightly more complex (and consequently slower) than using a rule list like CBA. Although the CMAR classifier is slower to make a prediction than the CBA classifier (considering the same number of rules in both classifiers), its worst-case time complexity is the same, i.e., $O(\eta|\mathcal{I}|)$.

#### *3.2.3. Principal Association Mining (PAM)*

*Principal Association Mining* (PAM) [15] also uses a modified version of Apriori [6] for the rule generation stage and all mined CARs are both frequent and strong. But contrary to what is done in CBA, PAM keeps all the conflicting rules, i.e, rules with the same condition but with different class labels are retained as candidate rules for the next stage. This feature differentiates PAM from more conventional ACs, which retain only the optimal rule for each condition. Note that to have conflicting rules, we must set $minconf \leq 50\%$.

For the classifier builder, PAM ranks the rules as follows. Given two rules $R_i$ and $R_j$, $R_i$ is prior to $R_j$ if:

1. $conf(R_i) > conf(R_j)$, or
2. $conf(R_i) = conf(R_j)$ and $sup(R_i) > sup(R_j)$, or
3. $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$ and $len(R_i) > len(R_j)$, or
4. $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$, $len(R_i) = len(R_j)$ and $lex(R_i) < lex(R_j)$,

where $len(R)$ denotes the number of items in rule $R$, and $lex(R)$ denotes the position of $R$ in the lexicographic order on items, respectively.

Next, PAM applies the following rule pruning. All rules are categorized into groups according to their class labels. Subsequently, the rules with lower confidence subsumed by more **specific** rules

with higher confidence will be pruned. Each group performs the pruning step independently so that conflicting rules can be kept. This rule pruning can be computationally expensive: $O(\kappa^2)$. The authors did not explain how to implement this rule pruning efficiently.

Thereafter, PAM also uses a heuristic based on the data set coverage to select the rules for the classifier, as shown in Algorithm 1. PAM uses a metric called *principality*, which is a convex combination between confidence and completeness: $princ(X \to c) = \lambda \cdot conf(X \to c) + (1 - \lambda) \cdot comp(X \to c)$. Different from other AC algorithms, PAM combines removal of training objects with the recursive recalculation of the quality (principality) of the remaining rules. For this reason, the heuristic used by PAM is computationally more expensive than the ones used by CBA and CMAR. Due to this recalculation, the time complexity of this heuristic is $O(\kappa_*^2|\mathcal{T}||\mathcal{I}|)$.

**Algorithm 1** PAM Classifier construction

**Input:** a set of rules $\mathcal{R}$ and the training data set $T$
**Output:** associative classifier $AC$

```
while R is not empty and T is not empty do
    Calculate the Principality for each rule in R;
    Select the rule with the highest Principality R*
    if R* classifies at least one object correctly then
        Add R* into AC
        Delete R* from R
        Delete all objects covered by R* from T
```

**Lemma 2.** *The condition of a CAR selected to the PAM's classifier is a closed itemset.*

*Proof.* Remark that each itemset $Y \subseteq \mathcal{I}$ belongs to only one equivalence class. Let $X$ be a closed itemset, only one itemset $Y \in Equiv(X)$ can be the condition of a selected CAR $Y \to c$ since the classifier builder removes all data records that are covered by the rule and a selected rule must correctly classify at least one data record. Among all possible CAR $Y \to c$ with $Y \in Equiv(X)$, the CAR $X \to c$ has the higher rank. □

The classifier of PAM is also a rule list with the form $\langle R_1, R_2, \cdots, default\_class \rangle$.

## 4. Proposed methods

### *4.1. Best Rule (BR)*

For the rule generator stage, the *Best Rule* (BR) can use any pattern mining algorithm that enumerates all **closed** frequent itemsets. For convenience, we adapted the RIn-Close_CVCP algorithm [55, 54] to mine frequent and strong CARs. RIn-Close_CVCP has time complexity $O(\mathfrak{c}|\mathcal{T}|m(\log|\mathcal{T}| + m))$, where $m$ is the number of attributes in the data set. It is an adaptation of the In-Close5 algorithm. RIn-Close_CVCP handles the discretized data directly, i.e., there is no need to binarize the data.

For the classifier builder, the (BR) classifier also uses the metric *principality*. It ranks the rules as follows. Given two rules $R_i$ and $R_j$, $R_i$ is prior to $R_j$ if:

1. $princ(R_i) > princ(R_j)$, or
2. $princ(R_i) = princ(R_j)$ and $conf(R_i) > conf(R_j)$, or
3. $princ(R_i) = princ(R_j)$ and $conf(R_i) = conf(R_j)$ and $sup(R_i) > sup(R_j)$, or
4. $princ(R_i) = princ(R_j)$ and $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$ and $len(R_i) > len(R_j)$, or
5. $princ(R_i) = princ(R_j)$ and $conf(R_i) = conf(R_j)$, $sup(R_i) = sup(R_j)$, $len(R_i) = len(R_j)$ and $lex(R_i) < lex(R_j)$.

To select the rules for the classifier, BR naively chooses the best rule for each class label. Thus, the computation cost to build the classifier is very low, and results in very few rules. Further, with very few rules, the time required to classify a new sample is also very low.

BR also uses a default class label in the classifier. So, after the selection of the rules, the classifier has the form: $\langle R_1, R_2, \cdots, R_{k-1}, default\_class \rangle$, where $k$ is the number of class labels in the data set. Thus, the classifiers provided by CBA, PAM and BR are rule lists with time complexity $O(\eta|\mathcal{I}|)$ for predicting an unseen instance. In this case $\eta = k$.

### *4.2. New Variations*

This section presents several variations of the algorithms presented in Sections 3.2.2, 3.2.3, and 4.1.

#### *4.2.1. Adding Default Class Label to CMAR (CMAR*)*

CMAR does not assign any class label to a test sample that does not match its rules. In order to address this limitation of CMAR, especially when it is applied to very unbalanced data sets, CMAR* is a variant of CMAR that assigns the default label to a test sample that does not match its rules. So, CMAR*'s classifier has the form: $\langle R_1, R_2, \cdots, default\_class \rangle$.

#### *4.2.2. Without Rule Pruning in PAM (-woG)*

Recall that PAM performs a rule pruning where all rules are categorized into groups according to their class labels and, subsequently, the rules with lower confidence subsumed by more specific rules with higher confidence are pruned. Due to the computation time taken to perform this pruning, we propose a variation that forgoes it. The suffix *-woG* is used to indicate a variation of a PAM-based algorithm that does not perform this pruning.

*4.2.3. RIn-Close in PAM and BR (-R)*

The PAM and BR algorithms have a variation that consists of using *RIn-Close_CVC* [54, 55] (denoted with the suffix -R) in the rule generator stage. RIn-Close_CVC generalizes a traditional algorithm called In-Close5 [8] that can enumerate all closed frequent itemsets efficiently.

Note that CBA and CMAR cannot take advantage of RIn-Close_CVC since they prioritizes general rules, unlike PAM-based and BR-based that prioritizes specific rules (see Lemmas 1 and 2).

Unlike traditional pattern mining algorithms, the RIn-Close_CVC algorithm handles directly non-binary attributes. Also, for the continuous attributes, RIn-Close_CVC performs an online partitioning instead of an a priori partitioning. Any a priori and feasible partitioning involves loss of information and guides to sub-optimal solutions when compared to online partitioning. Working directly on the numerical attributes, proposals such as RIn-Close_CVC perform an online checking for correctness at the time the patterns are being built [54].

When dealing with data sets that have only discrete attributes, the results of RIn-Close_CVC and a traditional algorithm are equivalent [54]. Thus, the differences among PAM, BR and their variants using RIn-Close_CVC are given by the a priori or online partitioning of the continuous attributes.

While the time complexity of In-Close5 is $O(\mathfrak{c}|\mathcal{T}||\mathcal{I}|^2)$, the time complexity of RIn-Close is $O(\mathfrak{c}|\mathcal{T}|^3|m^2)$, where $m$ is the number of attributes in the data set.

## 5. Data Sets

This section considers the data sets used in this work. These are divided into two groups, the training data sets and the validation data sets.

All the data sets used in this work have the same features. Each of these data sets contains the label (file hash) of the program considered, the 119 features extracted from this program, and the class label. The selection of features that are in the data sets is those that have been used to detect and/or classify packers in prior work. Collectively these features have been shown to be effective and accurate, however there is no consensus on which features or approaches are best [11, 12, 13, 17, 24, 26, 28, 36, 42, 44, 46, 50, 51, 57] Thus these features, and consequently data sets, are interesting for rule mining which may identify effective approaches and features.

Note that the data sets with complete feature information cannot be released for non-disclosure reasons. However, the meta-data of the data sets (file hashes and packer labels) for all the data sets used in this work are available at `https://github.com/dtgw/packing-data-sets/tree/master/association-rule-mining`.

The features are diverse in their type and their range. A brief overview of the features is provided in Table 1 (a full description is provided in Appendix B). Observe that many of the 119 attributes are

| # Features | Kind | Type | Range |
|---|---|---|---|
| 64 | byte | Integer | $0-255$ |
| 28 | naturals | Integer | $0-$ |
| 16 | boolean | Boolean | $0,1$ |
| 6 | entropy | Float | $0-8$ |
| 5 | ratios | Float | $0-$ |

Table 1: Features Overview.

continuous, although not all are treated so as here. For example, some of the natural values are based on checksum or file location addresses, which do not have a clear relation. However, 18 attributes are continuous and have a partition-width $\epsilon > 0$ associated with them, e.g. entropy values and counting of dangerous functions included. It is important information because these continuous attributes are responsible for the difference between the AC's results with/without RIn-Close algorithm in the rule generator stage (online or a priori partitioning of the continuous attributes).

The class label is obtained by using three YARA rules to obtain packer information. This is the standard approach and was here based upon a synthesis of multiple YARA rules [13].

The sample programs are taken from a daily feed that supplied curated malicious programs for research and experimentation [49]. The samples are available at a rate of 1000 samples per day, and were obtained at this rate here.

### *5.1. Data Preparation*

In this work, for the usage of traditional frequent pattern mining algorithms in the rule generator stage, each *item* is given by an attribute-value pair in the way described in Section 2. To perform the a priori partitioning of the continuous attributes, we used equal-width bins. The bin size $\epsilon_j$ for each continuous attribute $j$ was defined by the expert and the same values were used in the online partitioning of RIn-Close_CVC. Full details of the bin size for each continuous attribute are shown in Table 2. The values for the bin sizes (i.e. the $\epsilon_j$) were chosen by an expert after analysis of the attributes and their distributions. Note that attributes that do not appear in Table 2 were not partitioned.

### *5.2. Training Data Sets*

The training data sets are constructed to be used for training the rule mining algorithms. Since all the algorithms consider how many rows (or sample programs) are used for the rule construction, the training data sets should be balanced (i.e. all class labels equally represented). To achieve this a large *common training data set* was created that contained all the sample programs available from 2019-08-01 until 2019-12-31. The data set was analyzed to find how often each class label appeared. Then training data sets were constructed by picking a minimum number $n$ of rows to have. Once this

| # | **Feature** | $\epsilon_j$ |
|---|---|---|
| 13 | OS Minor version number | 2.00 |
| 22 | Number of standards sections PE holds | 2.00 |
| 23 | Number of non-standards sections PE holds | 1.00 |
| 24 | Ratio between number of standards sections found and number of all sections found in PE under analysis | 0.10 |
| 29 | Number of readable and writable sections | 1.00 |
| 37 | Ratio between raw data and virtual size for section of entry point | 0.10 |
| 39 | Number of sections having their virtual size greater than their raw data size | 1.00 |
| 40 | Maximum ratio raw data per virtual size among all sections | 20.00 |
| 41 | Minimum ratio raw data per virtual size among all sections | 0.10 |
| 43 | Entropy of Code/text sections | 0.25 |
| 44 | Entropy of data section | 0.25 |
| 45 | Entropy of resource section | 0.50 |
| 46 | Entropy of PE header | 0.25 |
| 47 | Entropy of entire PE file | 0.50 |
| 48 | Entropy of section holding Entry point (EP) of PE under analysis | 0.50 |
| 114 | Number of functions imported found in import table directory (IDT) | 1.00 |
| 115 | Number of malicious APIs imported | 2.00 |
| 116 | Ratio between number of malicious APIs imported to number of all functions imported by PE | 0.20 |

Table 2: Bin sizes for continuous attributes.

number was chosen, for each label that appeared at least $n$ times, $n$ randomly selected samples with the label were taken from the common training data set. These were then used as the training set with that number $n$.

For example, to construct the *balanced-75* training data set, 75 randomly selected samples were taken from the common training data set for each label that had at least 75 samples.

Aside: There are proposals in the literature that can handle minimum supports on a per label basis [15]. However, since not all algorithms considered here can, it is easier to compare by fixing the support and using balanced data sets.

### *5.3. Classification Data Sets*

The classification data sets are constructed only for validation and testing of the classifiers that have been trained. These data sets do not need to be balanced, and can also include labels that do not appear in the training data sets (due to these labels being infrequent in the common training data set). Each classification training data set is simply created by taking all the samples available during the specified time period.

For example, to construct the *classification-2019-12* data set, all the samples taken from 2019-12-01 to 2019-12-31 (inclusive) were added to this data set.

Observe that the test data sets can include labels that do not appear in the training data sets and so the classifier is not prepared for these unknown labels. To handle this, these unknown labels are not

considered in the evaluation of the algorithms (see Section 6).

## 6. Methodology

This section overviews the methodology used to evaluate the algorithms presented in Section 6.1 using the data sets presented in Section 5.

The methodology is applied in the same manner for each of the training data sets presented in Section 5.2. Observe that all the algorithms have two common parameters in the minimum confidence ($minconf$) and minimum support ($minsup$) to consider a rule. The parameter $minsup$ is instantiated to the values shown in Table 3. We set $minconf = 85\%$. For CMAR, we used $\gamma = 3.8415$ and $\delta = 3$, which are the default options in the implementation used. Lastly, based on the PAM results[15], we set $\lambda = 0.8$.

Then for each data set and parameter value, each algorithm is run and a classifier is constructed using the rules mined by that algorithm. These classifiers are then applied to each of the classification data sets described in 5.3.

Note that some training data sets have different class labels to the classification data sets (as discussed in Section 5.3).

The results of the classification are evaluated using *accuracy* ($ac$) that is calculated as follows

$$ac(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} 1(\hat{y}_i = y_i), \tag{3}$$

where $n_{samples}$ is the number of samples, $1(x)$ is the indicator function, $\hat{y}_i$ is the predicted value of the $i$-th sample and $y_i$ is the corresponding true value. Observe that this choice of equation does not consider the balance of the data set and uses the raw numbers for calculation.

An alternative *balanced accuracy* ($ba$) is also considered as follows

$$ba(y, \hat{y}, w) = \frac{1}{\sum \hat{w}_i} \sum_{i=1}^{n_{samples}} 1(\hat{y}_i = y_i)\hat{w}_i, \tag{4}$$

where $\hat{w}_i$ is the adjusted sample weight:

$$\hat{w}_i = \frac{1}{\sum_j 1(y_j = y_i)}. \tag{5}$$

Observe that the balanced accuracy modulates the weight of each result by the number of instances in the data set that has that label.

The accuracy calculation (Equation 3) therefore is used to indicate how effective the classifier is on a typical data set, while the balanced accuracy (Equation 4) indicates how well the classifier performs

when accounting for the frequency of the labels, and gives more emphasis on consistency over all labels (and not just common labels).

We also present a statistical significance test of the accuracy and balanced accuracy results. We use the nonparametric Wilcoxon rank sum test [23]. It tests if two independent samples $\mathbf{x}$ and $\mathbf{y}$ come from identical continuous (not necessarily normal) distributions with equal medians, against the alternative that they do not have equal medians. In fact, we used the right-tailed hypothesis test, where the alternative hypothesis states that the median of $\mathbf{x}$ is greater than the median of $\mathbf{y}$.

The experiments here used the implementations of CBA and CMAR available at the LUCS-KDD software library[1]. The CMAR* variation was a slight modification of the LUCS-KDD implementation. For the PAM-based algorithms (PAM, PAM-woG, PAM-R, and PAM-woG) and BR-based algorithms (BR, and BR-R) our own implementation written in Python was used. For RIn-Close, we used the implementation available at github repository[2]. Note that since the implementations are of different quality and from different sources, this makes runtime comparison unreasonable and so this has not been included in the analysis. However, commentary on computational complexity and required aspects that would impact the outcomes regardless of implementation are still discussed (see Section 8)

Lastly, in the Discussion section (Section 8), we compared the performance of the best AC algorithms presented in Section 6.1 against two other interpretable techniques, Decision Tree and RIPPER [18], and a non-interpretable technique, XGBoost [16]. This comparison intends to illustrate better the effectiveness of the associative classification algorithms in the problem of packer classification. For the Decision Tree, we used the scikit-learn implementation [41] and set $max\_leaf\_nodes = 14$ corresponding to the 14 label classes in the training set. For RIPPER, we use the R package *arulesCBA* [20] with the default setting. For XGBoost, we use the Python package XGBoost [1] with the following parameters: $max\_depth = 2$, $eta = 1$, $objective = multi : softmax$, $num\_class = 14$, $eval\_metric = merror$, $nthread = 4$, and $num\_round = 10$. The *objective* and *eval_metric* parameters were set for multi-class classification. The *num_round* parameter was set based on the stability of the training error, i.e., $merror = 0.000952$ from round 6 until 10.

### *6.1. Algorithms Summary*

The algorithms used in this work are as follows with novel algorithms in bold.

[1] `https://cgi.csc.liv.ac.uk/~frans/KDD/Software/`
[2] `https://github.com/rveroneze/rinclose`

| | |
|---|---|
| CBA | Classification based on associations from [33]. |
| CMAR | Classification based on multiple association rules from [30]. |
| **CMAR*** | Classification based on multiple association rules with a default class label as defined in Section 4.2.1. |
| PAM | Principal association mining as defined in [15]. |
| **PAM-woG** | PAM extended without rule pruning as defined in Section 4.2.2. |
| **PAM-R** | PAM extended with RIn-Close as defined in Section 4.2.3. |
| **PAM-R-woG** | PAM extended without rule pruning and with RIn-Close as defined in Sections 4.2.2 and 4.2.3. |
| **BR** | Best rule algorithm as defined in Section 4.1. |
| **BR-R** | BR extended with RIn-Close as defined in Section 4.2.3. |

## 7. Results

In this section, we first present the results of experimenting with all the algorithms and as described using the method from Section 6. Then, we present a statistical significance test of our results.

### *7.1. Experimental results*

An overview of the results for the algorithms with three different minimum supports (30, 45, and 60) on eight different testing and validation sets (listed according to the month from which the samples were collected in) and considering the accuracy can be seen in Table 3 and also Figures 1(a), 1(c) and 1(e). Note that in Table 3 the best result for a fixed *minsup* is indicated in **bold** and the best regardless of *minsup* in ***bold and italic***. Observe that the BR-R algorithm performs the best overall (best for 13 of the 24 combinations), closely followed by the BR algorithm (best for 9 of 24 combinations). The PAM-R algorithm also performs well, albeit with only a single best result for all the combinations. PAM also performs well, but the other variants of PAM (PAM-woG and PAM-R-woG) do not. Finally, the CMAR and CMAR* algorithms perform terribly in all combinations.

Of the eight data sets the best results were consistently achieved by the BR and BR-R algorithms (3 best results each, regardless of the *minsup* parameter). Interestingly the BR and BR-R algorithms appear to have no sensitivity to the *minsup* parameter and always achieved the same accuracy.

One interesting observation is that the accuracy drops off over time. This is a known challenge in the area of malware and packers since new variations and versions of programs appear over time and old ones (that can be effectively countered) become rarer. The results here indicate that all the algorithms except CBA with *minsup*=60 suffer from a reduction in accuracy over time, particularly

| | | Accuracy (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *minsup* | **Algorithm** | **2019-10** | **2019-11** | **2019-12** | **2020-01** | **2020-02** | **2020-03** | **2020-04** | **2020-05** |
| 30 | CBA | 93.14 | 94.01 | 93.68 | 94.10 | 90.93 | 91.60 | 92.07 | 92.21 |
| | CMAR | 12.35 | 16.84 | 18.66 | 14.27 | 10.43 | 10.77 | 14.26 | 18.21 |
| | CMAR* | 34.22 | 32.81 | 31.62 | 28.45 | 27.95 | 28.74 | 28.10 | 31.08 |
| | PAM | 96.64 | 97.72 | 97.55 | 97.08 | 96.06 | 97.79 | 97.06 | 94.31 |
| | PAM-woG | 90.76 | 89.75 | 92.06 | 92.18 | 91.95 | 91.16 | 91.31 | 87.91 |
| | PAM-R | 98.65 | ***99.08*** | 98.79 | 98.74 | 97.39 | 98.42 | 98.12 | 95.48 |
| | PAM-R-woG | 95.41 | 95.66 | 95.95 | 96.66 | 95.39 | 96.03 | 94.61 | 91.83 |
| | BR | 98.50 | 98.86 | 99.08 | ***99.23*** | ***98.63*** | 99.03 | ***98.70*** | 95.29 |
| | BR-R | ***98.79*** | 99.02 | ***99.23*** | 99.14 | 98.46 | ***99.15*** | 98.31 | **96.03** |
| 45 | CBA | 93.14 | 94.01 | 93.68 | 94.10 | 90.93 | 91.60 | 92.07 | 92.21 |
| | CMAR | 12.35 | 16.84 | 18.66 | 14.27 | 10.43 | 10.77 | 14.26 | 18.21 |
| | CMAR* | 34.73 | 33.47 | 31.90 | 29.11 | 27.92 | 29.07 | 27.34 | 33.08 |
| | PAM | 97.91 | 98.59 | 98.48 | 98.30 | 97.39 | 97.95 | 97.55 | 94.47 |
| | PAM-woG | 90.76 | 89.75 | 92.06 | 92.18 | 91.95 | 91.16 | 91.31 | 87.91 |
| | PAM-R | 98.51 | 99.00 | 98.90 | 98.89 | 97.61 | 98.71 | 98.46 | 95.99 |
| | PAM-R-woG | 95.41 | 95.66 | 95.95 | 96.66 | 95.39 | 96.03 | 94.61 | 91.83 |
| | BR | 98.50 | 98.86 | 99.08 | ***99.23*** | ***98.63*** | 99.03 | ***98.70*** | 95.29 |
| | BR-R | ***98.79*** | **99.02** | ***99.23*** | 99.14 | 98.46 | ***99.15*** | 98.31 | **96.03** |
| 60 | CBA | 96.80 | 97.24 | 96.94 | 96.71 | 94.89 | 95.97 | 96.49 | ***96.29*** |
| | CMAR | 12.28 | 16.81 | 18.65 | 14.26 | 10.36 | 10.74 | 14.25 | 18.20 |
| | CMAR* | 31.83 | 37.38 | 36.38 | 32.44 | 33.52 | 31.91 | 25.33 | 22.92 |
| | PAM | 91.76 | 93.49 | 94.82 | 94.92 | 93.65 | 94.30 | 92.45 | 87.93 |
| | PAM-woG | 89.34 | 88.56 | 90.96 | 91.19 | 90.92 | 90.24 | 89.37 | 86.68 |
| | PAM-R | 96.37 | 96.90 | 97.03 | 97.29 | 95.82 | 96.57 | 95.38 | 92.49 |
| | PAM-R-woG | 95.41 | 95.66 | 95.95 | 96.66 | 95.39 | 96.03 | 94.61 | 91.83 |
| | BR | 98.50 | 98.86 | 99.08 | ***99.23*** | ***98.63*** | 99.03 | ***98.70*** | 95.29 |
| | BR-R | ***98.79*** | **99.02** | ***99.23*** | 99.14 | 98.46 | ***99.15*** | 98.31 | 96.03 |

Table 3: Accuracy Results.

five months after the end of the training data. This indicates that an effective classifier should be retrained within this time period.

An overview of the results for the algorithms with different supports on eight testing and validation sets (again listed according to the month from which the samples were collected in) and considering the balanced accuracy can be seen in Table 4 and Figures 1(b), 1(d) and 1(f). Observe that here the balanced accuracy is more consistent between all the algorithms, with even CMAR and CMAR* performing reasonably. The best results appear to be from PAM-based algorithms, with: PAM being best for 4 combinations; PAM-woG being best for 6 combinations; PAM-R being best for 3 combinations; and PAM-R-woG being best for 4 combinations. For all the PAM-based algorithms being best for 17 of the 24 combinations. The remaining 7 best results were all achieved by the CBA algorithm. Observe that the BR, BR-R, CMAR, and CMAR* algorithms all perform consistently worse than the CBA and PAM-based algorithms. Finally, observe that all algorithms degrade in performance three months after the conclusion of the training sets (i.e. for 2020-04 and 2020-05).

Of the eight data sets the best results overall were distributed as follows: CBA scored best for

| | | Balanced Accuracy (%) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *minsup* | **Algorithm** | **2019-10** | **2019-11** | **2019-12** | **2020-01** | **2020-02** | **2020-03** | **2020-04** | **2020-05** |
| 30 | CBA | 97.40 | 95.43 | ***97.66*** | 95.65 | ***97.75*** | 96.12 | 89.58 | ***88.41*** |
| | CMAR | 90.94 | 88.80 | 91.07 | 89.00 | 91.32 | 89.18 | 83.15 | 82.32 |
| | CMAR* | 92.72 | 90.17 | 92.21 | 90.18 | 92.72 | 90.72 | 84.31 | 83.53 |
| | PAM | 97.16 | 94.80 | 95.03 | 96.25 | 91.92 | **98.67** | 89.48 | 83.47 |
| | PAM-woG | 97.75 | 97.36 | 95.98 | **96.86** | 97.11 | 98.10 | ***92.57*** | 86.28 |
| | PAM-R | 96.65 | 96.31 | 91.14 | 89.09 | 91.99 | 87.28 | 82.72 | 79.81 |
| | PAM-R-woG | **98.17** | **98.25** | 96.25 | 93.11 | 93.02 | 87.07 | 82.43 | 81.18 |
| | BR | 95.59 | 92.14 | 95.03 | 95.96 | 91.68 | 90.47 | 86.36 | 80.47 |
| | BR-R | 93.93 | 91.11 | 91.05 | 87.64 | 91.62 | 79.07 | 79.48 | 78.67 |
| 45 | CBA | 97.40 | 95.43 | ***97.66*** | 95.65 | ***97.75*** | 96.12 | 89.58 | ***88.41*** |
| | CMAR | 90.94 | 88.80 | 91.07 | 89.00 | 91.32 | 89.18 | 83.15 | 82.32 |
| | CMAR* | 92.76 | 90.23 | 92.23 | 90.24 | 92.71 | 90.75 | 84.24 | 83.72 |
| | PAM | 96.55 | 94.88 | 95.11 | 96.35 | 92.03 | ***98.69*** | 89.52 | 83.48 |
| | PAM-woG | 97.75 | 97.36 | 95.98 | **96.86** | 97.11 | 98.10 | ***92.57*** | 86.28 |
| | PAM-R | 95.21 | 93.71 | 91.15 | 88.08 | 92.00 | 87.30 | 82.75 | 79.86 |
| | PAM-R-woG | **98.17** | **98.25** | 96.25 | 93.11 | 93.02 | 87.07 | 82.43 | 81.18 |
| | BR | 95.59 | 92.14 | 95.03 | 95.96 | 91.68 | 90.47 | 86.36 | 80.47 |
| | BR-R | 93.93 | 91.11 | 91.05 | 87.64 | 91.62 | 79.07 | 79.48 | 78.67 |
| 60 | CBA | 91.42 | 91.81 | 90.80 | 91.79 | 91.25 | 88.81 | 86.37 | **87.51** |
| | CMAR | 84.49 | 84.10 | 83.93 | 84.92 | 84.51 | 81.40 | 79.58 | 80.16 |
| | CMAR* | 86.08 | 85.86 | 85.48 | 86.44 | 86.35 | 83.23 | 80.50 | 80.60 |
| | PAM | 97.83 | 97.68 | 96.22 | ***97.09*** | 93.68 | **98.37** | 89.10 | 84.15 |
| | PAM-woG | 97.64 | 97.26 | 95.88 | 96.78 | **97.03** | 98.02 | **92.41** | 86.17 |
| | PAM-R | ***98.25*** | ***98.36*** | **96.34** | 93.16 | 93.05 | 87.12 | 82.49 | 81.24 |
| | PAM-R-woG | 98.17 | 98.25 | 96.25 | 93.11 | 93.02 | 87.07 | 82.43 | 81.18 |
| | BR | 95.59 | 92.14 | 95.03 | 95.96 | 91.68 | 90.47 | 86.36 | 80.47 |
| | BR-R | 93.93 | 91.11 | 91.05 | 87.64 | 91.62 | 79.07 | 79.48 | 78.67 |

Table 4: Balanced Accuracy Results.

3 data sets; PAM best for 2 data sets; PAM-woG best for 1 data set; and PAM-R best for 2 data sets. Interestingly for CBA when considering the balanced accuracy, the *minsup* acts conversely to when considering accuracy: lower *minsup* is better for balanced accuracy. For PAM-based algorithms, there is a minor impact from variations of the *minsup* parameter (except for PAM-R-woG which is unaffected). The CMAR and CMAR* algorithms performed slightly better on balanced accuracy when the *minsup* was lower. Finally, the BR and BR-R algorithms remain unaffected by the *minsup* parameter as before.

As for the accuracy, the balanced accuracy also drops off over time. This is as expected, but also much more significant in the results here (as best illustrated in Figures 1).

An overview of the number of rules and number of attributes used in the classifier for each algorithm and support is shown in Table 5, with detailed discussion in Section 8.

### *7.2. Statistical significance test of the results*

Table 6 shows the $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the accuracy results considering the different values of the parameter *minsup* used in our experiments.

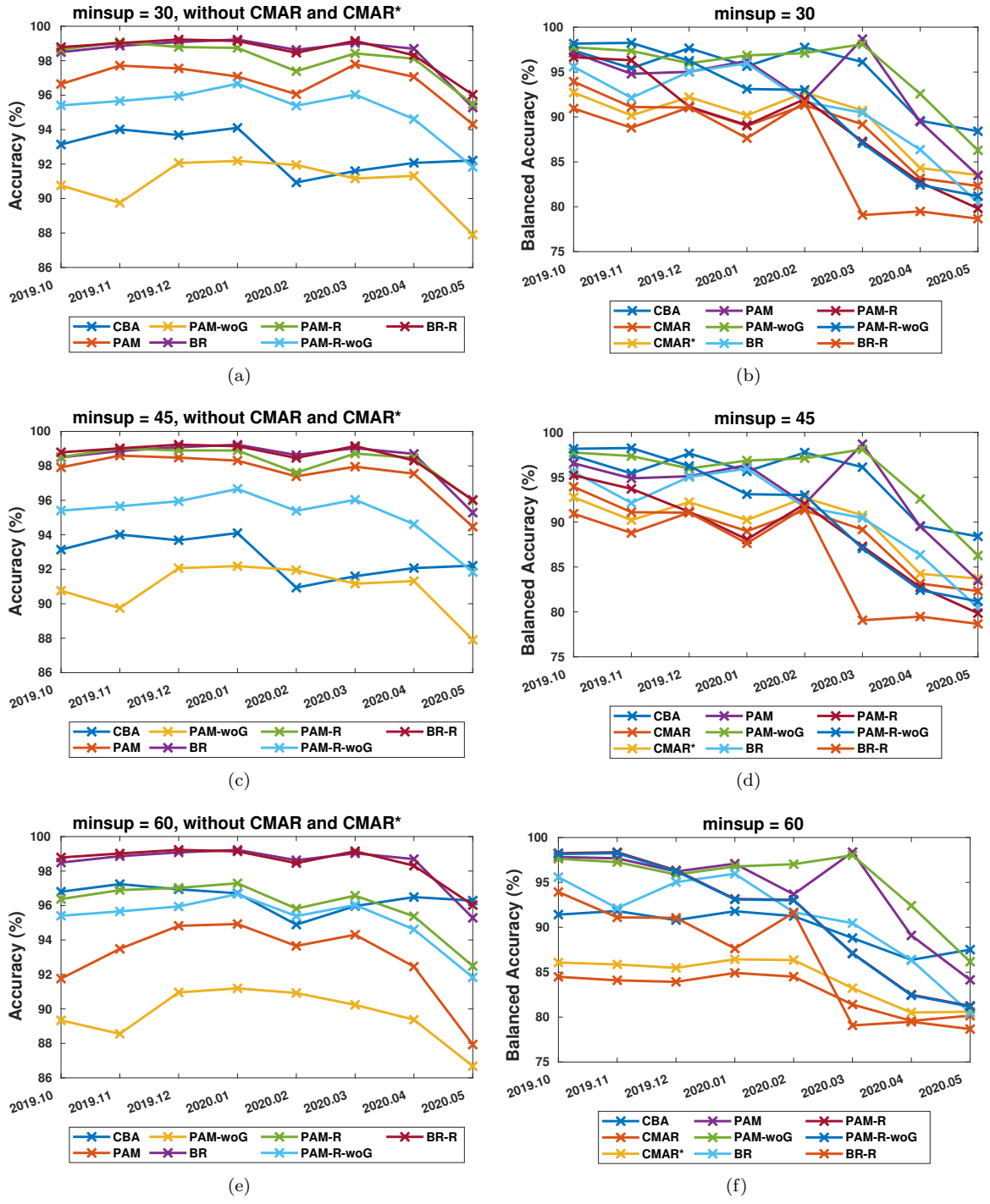


Figure 1: Accuracy and Balanced Accuracy Graphs.

CBA with $minsup = 60$ was superior than CBA with $minsup = 30$ or $minsup = 45$. PAM with $minsup = 45$ was superior than PAM with $minsup = 30$ or $minsup = 60$. PAM-woG with $minsup = 60$

| *minsup* | **Algorithm** | **#Rules** | **#Attributes** |
|---|---|---|---|
| 30 | CBA | 22 | 27 |
| | CMAR | 361 | 106 |
| | CMAR* | 361 | 106 |
| | PAM | 21 | 116 |
| | PAM-woG | 27 | 117 |
| | PAM-R | 24 | 117 |
| | PAM-R-woG | 24 | 115 |
| | BR | 13 | 116 |
| | BR-R | 13 | 115 |
| 45 | CBA | 22 | 27 |
| | CMAR | 358 | 105 |
| | CMAR* | 358 | 105 |
| | PAM | 19 | 116 |
| | PAM-woG | 27 | 117 |
| | PAM-R | 20 | 117 |
| | PAM-R-woG | 24 | 115 |
| | BR | 13 | 116 |
| | BR-R | 13 | 115 |
| 60 | CBA | 15 | 18 |
| | CMAR | 508 | 107 |
| | CMAR* | 508 | 107 |
| | PAM | 23 | 116 |
| | PAM-woG | 27 | 116 |
| | PAM-R | 25 | 115 |
| | PAM-R-woG | 24 | 115 |
| | BR | 13 | 116 |
| | BR-R | 13 | 115 |

Table 5: Number of rules and attributes used in the classifiers.

was inferior than PAM-woG with $minsup = 30$ or $minsup = 45$ (the same for PAM-R). Finally, CMAR, CMAR*, PAM-R-woG, BR and BR-R had negligible sensitivity to *minsup*.

Considering these results, our further statistical significance tests for accuracy considers $minsup = 60$ for CBA and $minsup = 45$ for the other algorithms. Table 7 shows the $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the accuracy results considering the different algorithms used in our experiments. Table 8 summarizes the results of Table 7.

PAM-R, BR and BR-R were significantly better than 6 algorithms (CBA, CMAR, CMAR*, PAM, PAM-woG and PAM-R-woG) and were not significantly worse than any algorithm. Thus, besides the fact that the PAM-R algorithm had the best accuracy just once, its performance in terms of accuracy was statistically equivalent to the performance of the BR and BR-R algorithms. PAM was inferior only to these 3 algorithms (PAM-R, BR and BR-R) and superior to the other 5 (CBA, CMAR, CMAR*, PAM-woG and PAM-R-woG).

Table 9 shows the $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the balanced accuracy results considering the different values of the parameter *minsup* used in our experiments. CBA with $minsup = 60$ was inferior than CBA with $minsup = 30$ or $minsup = 45$ (the

| | $30 > 45$ | $30 > 60$ | $45 > 30$ | $45 > 60$ | $60 > 30$ | $60 > 45$ |
|---|---|---|---|---|---|---|
| CBA | 0.5413 | 1.0000 | 0.5413 | 1.0000 | **0.0001** | **0.0001** |
| CMAR | 0.5413 | 0.3329 | 0.5413 | 0.3329 | 0.6875 | 0.6875 |
| CMAR* | 0.6773 | 0.8359 | 0.3605 | 0.7131 | 0.1911 | 0.3227 |
| PAM | 0.9824 | **0.0003** | **0.0204** | **0.0003** | 0.9998 | 0.9998 |
| PAM-woG | 0.5413 | **0.0415** | 0.5413 | **0.0415** | 0.9675 | 0.9675 |
| PAM-R | 0.7131 | **0.0023** | 0.3227 | **0.0015** | 0.9985 | 0.9991 |
| PAM-R-woG | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |
| BR | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |
| BR-R | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |

Table 6: $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the accuracy results considering the different values of the parameter *minsup* used in our experiments. $30 > 45$ stands for the alternative hypothesis that the median of the accuracy for $minsup = 30$ is greater than the median of the accuracy for $minsup = 45$. Equivalently for the other columns of the table.

| | CBA | CMAR | CMAR* | PAM | PAM-R-woG | PAM-R | PAM-R-woG | BR | BR-R |
|---|---|---|---|---|---|---|---|---|---|
| CBA | 0.5413 | **0.0001** | **0.0001** | 0.9965 | **0.0001** | 0.9985 | **0.0103** | 0.9977 | 0.9985 |
| CMAR | 1.0000 | 0.5413 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| CMAR* | 1.0000 | **0.0001** | 0.5413 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| PAM | **0.0052** | **0.0001** | **0.0001** | 0.5413 | **0.0001** | 0.9675 | **0.0035** | 0.9965 | 0.9897 |
| PAM-woG | 1.0000 | **0.0001** | **0.0001** | 1.0000 | 0.5413 | 1.0000 | 0.9997 | 1.0000 | 1.0000 |
| PAM-R | **0.0023** | **0.0001** | **0.0001** | **0.0415** | **0.0001** | 0.5413 | **0.0003** | 0.8089 | 0.8670 |
| PAM-R-woG | 0.9926 | **0.0001** | **0.0001** | 0.9977 | **0.0005** | 0.9998 | 0.5413 | 0.9985 | 0.9999 |
| BR | **0.0035** | **0.0001** | **0.0001** | **0.0052** | **0.0001** | 0.2209 | **0.0023** | 0.5413 | 0.5737 |
| BR-R | **0.0023** | **0.0001** | **0.0001** | **0.0141** | **0.0001** | 0.1456 | **0.0002** | 0.4521 | 0.5413 |

Table 7: $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the accuracy results considering the different algorithms used in our experiments. It considers $minsup = 60$ for CBA and $minsup = 45$ for the other algorithms. Each cell in the table indicates whether the accuracy median of the algorithm in line $i$ is greater than the accuracy median of the algorithm in column $j$.

| | Better than | Worse than |
|---|---|---|
| CBA | 4 | 4 |
| CMAR | 0 | 8 |
| CMAR* | 1 | 7 |
| PAM | 5 | 3 |
| PAM-woG | 2 | 6 |
| PAM-R | **6** | **0** |
| PAM-R-woG | 3 | 5 |
| BR | **6** | **0** |
| BR-R | **6** | **0** |

Table 8: Summary of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the accuracy results considering the different algorithms used in our experiments.

same for CMAR and CMAR*). There is no statistically significant difference among the values of *minsup* for the other algorithms. Thus, we chose $minsup = 45$ for our further statistical significance tests considering the balanced accuracy. Table 10 shows the $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the balanced accuracy results considering the different algorithms used in our experiments. Table 11 summarizes the results of Table 10.

CBA and PAM-woG were significantly better than 5 algorithms (CMAR, CMAR*, PAM-R, BR and BR-R) and were not significantly worse than any algorithm. Also, PAM and PAM-R-woG were not

significantly worse than any algorithms, with PAM being better than 3 algorithms (CMAR, PAM-R and BR-R). PAM-R were significantly worse than CBA and its variants PAM and PAM-woG. CMAR and BR-R were also significantly worse than CBA, PAM and PAM-woG. CMAR* and BR were significantly worse than CBA and PAM-woG.

| | $30 > 45$ | $30 > 60$ | $45 > 30$ | $45 > 60$ | $60 > 30$ | $60 > 45$ |
|---|---|---|---|---|---|---|
| CBA | 0.5413 | **0.0141** | 0.5413 | **0.0141** | 0.9897 | 0.9897 |
| CMAR | 0.5413 | **0.0103** | 0.5413 | **0.0103** | 0.9926 | 0.9926 |
| CMAR* | 0.6282 | **0.0101** | 0.3875 | **0.0103** | 0.9925 | 0.9926 |
| PAM | 0.6395 | 0.7131 | 0.3992 | 0.7473 | 0.3227 | 0.2869 |
| PAM-woG | 0.5413 | 0.3605 | 0.5413 | 0.3605 | 0.6773 | 0.6773 |
| PAM-R | 0.5204 | 0.7791 | 0.5204 | 0.8089 | 0.2527 | 0.2209 |
| PAM-R-woG | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |
| BR | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |
| BR-R | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 | 0.5413 |

Table 9: $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the balanced accuracy results considering the different values of the parameter *minsup* used in our experiments. $30 > 45$ stands for the alternative hypothesis that the median of the balanced accuracy for $minsup = 30$ is greater than the median of the balanced accuracy for $minsup = 45$. Equivalently for the other columns of the table.

| | CBA | CMAR | CMAR* | PAM | PAM-R-woG | PAM-R | PAM-R-woG | BR | BR-R |
|---|---|---|---|---|---|---|---|---|---|
| CBA | 0.5413 | **0.0074** | **0.0190** | 0.2527 | 0.6882 | **0.0052** | 0.2527 | **0.0415** | **0.0052** |
| CMAR | 0.9948 | 0.5413 | 0.8828 | 0.9948 | 0.9985 | 0.6773 | 0.8828 | 0.9476 | 0.4796 |
| CMAR* | 0.9859 | 0.1393 | 0.5413 | 0.9476 | 0.9965 | 0.4392 | 0.8607 | 0.8089 | 0.2527 |
| PAM | 0.7791 | **0.0074** | 0.0652 | 0.5413 | 0.9026 | **0.0325** | 0.2869 | 0.1641 | **0.0103** |
| PAM-woG | 0.3336 | **0.0023** | **0.0052** | 0.1172 | 0.5413 | **0.0052** | 0.2209 | **0.0103** | **0.0023** |
| PAM-R | 0.9965 | 0.3605 | 0.6008 | 0.9751 | 0.9965 | 0.5413 | 0.8089 | 0.8359 | 0.1911 |
| PAM-R-woG | 0.7791 | 0.1393 | 0.1641 | 0.7473 | 0.8089 | 0.2209 | 0.5413 | 0.2869 | 0.0652 |
| BR | 0.9675 | 0.0652 | 0.2209 | 0.8607 | 0.9926 | 0.1911 | 0.7473 | 0.5413 | 0.0524 |
| BR-R | 0.9965 | 0.5608 | 0.7791 | 0.9926 | 0.9985 | 0.8359 | 0.9476 | 0.9585 | 0.5413 |

Table 10: $p$-values of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the balanced accuracy results considering the different algorithms used in our experiments. It considers $minsup = 45$ for all algorithms. Each cell in the table indicates whether the balanced accuracy median of the algorithm in line $i$ is greater than the balanced accuracy median of the algorithm in column $j$.

| | Better than | Worse than |
|---|---|---|
| CBA | **5** | **0** |
| CMAR | 0 | 3 |
| CMAR* | 0 | 2 |
| PAM | 3 | **0** |
| PAM-woG | **5** | **0** |
| PAM-R | 0 | 3 |
| PAM-R-woG | 0 | **0** |
| BR | 0 | 2 |
| BR-R | 0 | 3 |

Table 11: Summary of the right-tailed Wilcoxon rank sum hypothesis test for the comparison of the balanced accuracy results considering the different algorithms used in our experiments.

## 8. Discussion

This section discusses how the algorithms compare in practicalities and some broader considerations from this work. This is focused on three aspects: the number of rules created for a classifier, the number of attributes used, and the classifier performance. The most suitable algorithms for packer classification considering performance, interpretability and computational cost are identified. We also comment on how to handle malware evolving.

### *8.1. Number of Classifier Rules*

The BR and BR-R algorithms used fewer rules in their classifiers because they use only one rule per class label. This leads to them having no sensitivity to the parameter *minsup* as shown in the results. However, note that if we use a very high *minsup* it may not be possible to find rules with the desired confidence. By contrast, the other algorithms use a heuristic based on the training sample coverage to select the rules to the classifier.

The CMAR and CMAR* algorithms stand out for using many more rules than the other methods. This happens because each training sample has to be satisfied (covered) by at least $\delta$ rules before it is ignored in the classifier builder. With $minsup = 60$, CMAR and CMAR* select significantly more rules in an attempt to cover the training samples $\delta$ times.

The CBA algorithm is comparable with the PAM-based algorithms, but selects fewer rules with $minsup = 60$ compared to $minsup \in \{30, 45\}$. The rules selected by the classifier builder with $minsup = 60$ are a subset of the selected rules with $minsup = 30$ and $minsup = 45$ (which are exactly the same). This indicates that some of the rules selected with $minsup = 30$ and $minsup = 45$ are not available to the classifier builder when using $minsup = 60$, and there are no other rules capable of replacing them. However, note that the LUCKS KDD implementation of CBA and CMAR used here has thresholds to limit the search from exploring every possible rule. Thus it may be possible to find suitable rules with $minsup = 60$, but these may take significant time and computation to discover.

For PAM-woG and PAM-R-woG, the number of selected rules are the same for all *minsup* values, but this does not hold for PAM and PAM-R. From the pattern mining literature, we know that if an itemset $X$ is frequent, all its subsets are also frequent and have support greater than or equal to $sup(X)$. So, without the pruning of the more general rules, we are more likely to select rules that cover samples not yet covered by their more specific versions. This makes the variants of PAM without the rule pruning less sensitive to the *minsup*.

### *8.2. Number of Attributes Used in Rules*

Due to the algorithmic designs, it is expected that all of PAM, PAM-woG, PAM-R, PAM-R-woG, BR and BR-R used many attributes in their classifiers since they select rules whose conditions are

closed itemsets. For similar algorithmic reasons CBA selects minimal generators and hence used much fewer attributes in its classifiers. Despite CMAR and CMAR* prioritize rules with few attributes, they use many rules, and so they used many attributes as well. Therefore, among the analyzed methods, CBA generated the most interpretable classifiers.

### *8.3. Classifier Performance*

An important feature of the malware data sets is that they are very unbalanced. The *not packed* class label comprises about 81% to 90% of the samples in the test data sets used in this work. Thus, by assigning *not packed* to all samples of a test data set, it is possible to reach 90% accuracy (but not balanced accuracy). Also, an effective strategy in terms of accuracy would be to assign a class label other than *not packed* only when you are very confident of it.

Thus, by selecting only the best rule per class label, BR and BR-R benefit from this feature of the malware data sets, promoting very good accuracies. Note also that the best accuracies produced by CBA is when it uses few rules (with $minsup = 60$). On the other hand, this feature is destructive to the CMAR and CMAR* poor accuracies.

The classifier builder computational costs of PAM and PAM-R are considerably higher than, respectively, the ones of PAM-woG and PAM-R-woG, but at least they produced significantly better accuracies than its -woG variants. With the pruning, PAM and PAM-R were able to select rules with less recall, but higher confidence (precision), which had a positive effect on accuracy. However, PAM and PAM-R performed numerically worse (not significantly statistically worse) than, respectively, PAM-woG and PAM-R-woG in terms of balanced accuracy. It indicates that, when all samples have the same weight to the performance computation, the more precise rules of PAM and PAM-R could have lower generalizability than the rules of PAM-woG and PAM-R-woG.

The usage of online partitioning of the numerical attributes can produce more specific rules than the a priori partitioning. The significantly better accuracies of PAM-R and PAM-R-woG when compared to, respectively, PAM and PAM-woG indicate that this was also beneficial for the accuracy. Although this effect was less and not statistically significant, BR-R had also slightly better accuracies than BR. But again we do not observe the same tendency for balanced accuracy. Indeed, PAM performed statistically significantly better than PAM-R.

This contrast between the results for accuracy and balanced accuracy is also observed in the results for CBA (which uses few attributes) with $minsup = 30$ and $minsup = 45$. While CBA's accuracy is easily outperformed by the methods that select closed itemsets, such as BR-R and BR, CBA's balanced accuracy is among the best results.

### 8.4. Overall

Considering the statistical significance test of our results regarding accuracy, PAM-R, BR and BR-R using $minsup = 45$ were the best options. The rule mining stage of these three algorithms is based on closed itemsets. But the rule mining stage of PAM-R and BR-R uses RIn-Close with online partitioning of the continuous features. Thus, the rule mining stage of PAM-R and BR-R is slower than that of BR. The classifier build stage of BR and BR-R is the same, and it is much simpler and faster than that of PAM-R. For these reasons, BR is faster than BR-R and PAM-R.

These three algorithms are equivalent in terms of the number of conditions in each rule. However, in BR and BR-R, the number of rules in the classifier is bounded by the number of class labels. Hence, PAM-R will usually produce a classifier with more rules than BR and BR-R. Consequently, the classifiers produced by BR and BR-R will usually be faster in making a prediction and more interpretable than that of PAM-R.

Regarding balanced accuracy, CBA and PAM-woG using $minsup = 45$ were the best options. The rule mining of PAM-woG can be done with algorithms for mining closed itemsets. This is an advantage in terms of computational cost. To avoid impractical runtimes, the CBA implementation stops the mining process after the number of frequent itemsets exceeds a defined threshold. It could be done for PAM-woG, but the number of closed frequent itemsets in the packer training dataset is tractable. To illustrate this information, using $minsup = 45$, the number of frequent itemsets **with length up to 3** (remark that the datasets have 119 features and more than 40,000 items after binarization) is equal to 1,499,915, while the number of all closed frequent itemsets is equal to 1,061,017. On the other hand, the classifier build stage of PAM-woG is slower than that of CBA due to the recalculation of the principality metric. Thus, the conclusion is that CBA will run faster than PAM-woG as long as the rule mining stage is stopped after exceeding a limit threshold for the number of frequent itemsets (our current evidence points out that short rules are effective for balanced accuracy). Besides, the more general rules used by CBA are more easily interpreted. Since the number of rules in the classifiers produced by CBA and PAM-woG are equivalent, CBA is definitely the best option when considering interpretability.

To better illustrate the effectiveness of these AC algorithms in the problem of packer classification, we make a comparison with other two rule-based techniques, i.e., Decision Tree (DT) and RIPPER [20], and a non-interpretable technique, i.e., XGBoost [1]. XGBoost is intended to provide performance baselines on what black-box models can achieve on these datasets. The results are reported in Tables 12 and 13 for accuracy and balanced accuracy, respectively. The best results among the interpretable classifiers are indicated in bold, and the best results overall are indicated in italic. The models produced by the DT and RIPPER algorithms are presented in Appendix C.

Observe that the accuracy of the algorithms considered here outperformed the other two interpretable classifiers, and even XGBoost, for all test data sets. For balanced accuracy, XGBoost is the best one for 5 of the 8 test data sets. However, the interpretable classifiers were able to achieve good balanced accuracies compared to XGBoost. Among the interpretable classifiers, the AC algorithms produced the best balanced accuracies for 5 data sets (4 for PAM-woG and 1 for CBA), while RIPPER produced the best balanced accuracies for the other 3 data sets.

Since BR, BR-R, and CBA using $minsup = 45$ are the best AC algorithms in terms of performance and interpretability, we also present the models produced by them in Appendix C. The main observation from these is that BR and BR-R build complex rules that consider many features, while CBA builds very simple rules. Despite this superficial observation, the detail of the rules and the common features can be used to identify which features are most significant in identifying the packer. Further, by identifying feature values that are related to packing (and not unpacked samples) can be used for detection [13]. This information can in turn help with further detection and classification algorithms, analysis, and research, as well as be applied to reverse engineering or generic systems.

| | **Accuracy (%)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Algorithm** | **2019-10** | **2019-11** | **2019-12** | **2020-01** | **2020-02** | **2020-03** | **2020-04** | **2020-05** |
| PAM-R | 98.51 | 99.00 | 98.90 | 98.89 | 97.61 | 98.71 | 98.46 | 95.99 |
| BR | 98.50 | 98.86 | 99.08 | ***99.23*** | ***98.63*** | 99.03 | ***98.70*** | 95.29 |
| BR-R | ***98.79*** | ***99.02*** | ***99.23*** | 99.14 | 98.46 | ***99.15*** | 98.31 | ***96.03*** |
| DT[3] | 60.40 | 65.63 | 70.28 | 71.14 | 72.77 | 73.81 | 59.04 | 54.50 |
| RIPPER[4] | 77.56 | 82.41 | 86.68 | 85.39 | 83.48 | 84.11 | 81.17 | 75.99 |
| XGBoost[5] | 82.49 | 89.20 | 92.31 | 92.79 | 91.07 | 91.98 | 85.38 | 81.83 |

Table 12: Accuracy results of other algorithms vs. PAM-R, BR and BR-R.

| | **Balanced Accuracy (%)** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Algorithm** | **2019-10** | **2019-11** | **2019-12** | **2020-01** | **2020-02** | **2020-03** | **2020-04** | **2020-05** |
| CBA | 97.40 | 95.43 | 97.66 | 95.65 | ***97.75*** | 96.12 | 89.58 | 88.41 |
| PAM-woG | **97.75** | 97.36 | 95.98 | **96.86** | 97.11 | **98.10** | ***92.57*** | 86.28 |
| DT | 91.87 | 91.19 | 94.22 | 92.77 | 90.77 | 85.70 | 80.43 | 79.54 |
| RIPPER | 97.65 | **98.01** | **98.82** | 96.70 | 97.05 | 95.39 | 88.04 | ***90.91*** |
| XGBoost | *98.08* | *98.79* | *99.25* | *97.25* | 95.10 | *99.08* | 92.50 | 90.58 |

Table 13: Balanced accuracy results of other algorithms vs. CBA and PAM-woG.

### *8.5. Malware evolving*

One area to consider in the domain of packer (and malware) detection and classification is the adversarial and evolving environment. Packers that are effectively detected and unpacked (or reverse engineered) then become ineffective for malware delivery and new packers are created to replace them. This in turn means that creating packer classifiers is not a one off exercise, instead the classifier must be retrained and updated regularly to meet the challenges of new packer varieties.

This then favors classifiers with a low cost to be built, which is the case of BR and BR-R. Actually, BR is the fastest AC algorithm studied here, followed by BR-R. Conveniently, these two algorithms also achieved the best performance in terms of accuracy. Besides their apparent degradation in the last month (2020-05), their results are still among the best. Thus, considering the cost to create the classifier modulated by the time it is effective, BR and BR-R (especially BR) are the best options in terms of accuracy.

For balanced accuracy, all algorithms' degradation over time is evident one month before (since 2020-04). Thus, if the main goal is to perform well when accounting for the frequency of the labels, then the classifiers must be retrained more frequently. In this case, CBA is yet the best choice as long as the threshold in the number of patterns does not affect the performance.

Instead of retraining the associative classifier from scratch, an option to be investigated is the incremental update of the rules. There are already several works in the literature for incremental/stream pattern mining [27, 32]. In addition, some works tackle the problem of associative classification over data streams [45].

## 9. Conclusions

Detecting packer programs is a significant challenge in cyber security and one that has seen many possible approaches used. Here we explore the effectiveness of nine variations of association-rule-based algorithms for packer classification on eight data sets consisting of $\sim 240,000$ real-world samples. The results here indicate that PAM-based algorithms perform well on data sets when accounting for both absolute and balanced accuracy. For absolute accuracy, the novel algorithms presented here BR and BR-R perform best overall. For balanced accuracy, CBA and the proposed PAM-woG were the best ones, followed by PAM. Also, the proposed PAM-R-woG was not statistically inferior to any competitor. The results also indicate that accuracy in all scenarios and for all algorithms degrades over time, indicating the regular updating of the classifier is vital to maintaining high accuracy.

Overall the novel algorithms and variations here show improvements in the classification of packers over the original algorithms. These improvements may also be suited to other applications where there are many features with different types, or when classification is significant.

Future work can build on this paper in many directions relating to algorithms, data sets, and related approaches.

Regarding algorithmic future work, one direction would be to explore further novel algorithms and improvements that may perform better on packing classification and similar data sets. Another direction would be to extend the algorithms, and, in particular, the rule and feature choices, to

consider more information such as feature extraction cost since not all features need to be extracted in applications like packer detection.

Regarding the data sets, one direction would be to explore the features used in more detail and examine which are most significant for classification, and also which can be omitted while maintaining good outcomes. Another would be to consider different approaches to training, e.g. using larger sets with smaller or larger numbers of samples in the training phase, and considering options like unbalanced training data sets. Yet another direction would be to consider longer-term longitudinal studies to evaluate the economics and evolution over a great period, including to validate economical predictions [13].

Finally, there are also related works that attempt to address the same challenges in a different approach such as machine learning algorithms like decision trees and PRISM [14]. A detailed comparison of the efficacy and comprehensibility of these approaches would also be of interest.

## Acknowledgments

R. Veroneze would like to thank FAPESP (Grants No. 2017/21174-8 and 2020/00123-9) for the financial support.

| | $a$ | $b$ | $c$ | $d$ | $l$ |
|---|---|---|---|---|---|
| 1 | 1.5 | 0.2 | 0.1 | 0.3 | 1 |
| 2 | 1.5 | 0.0 | 0.4 | 0.4 | 1 |
| 3 | 1.4 | 0.9 | 0.8 | 0.4 | 1 |
| 4 | 0.9 | 1.2 | 1.4 | 0.2 | 1 |
| 5 | 0.9 | 1.3 | 1.4 | 0.3 | 0 |
| 6 | 0.6 | 1.4 | 0.2 | 0.3 | 0 |
| 7 | 0.3 | 1.4 | 1.3 | 0.7 | 0 |

Table A.14: An example data set with four continuous attributes ranging from 0 to 1.5 and labels 0 or 1.

| | $a$ | $b$ | $c$ | $d$ | $l$ |
|---|---|---|---|---|---|
| 1 | 3 | 1 | 1 | 1 | 1 |
| 2 | 3 | 1 | 1 | 1 | 1 |
| 3 | 3 | 2 | 2 | 1 | 1 |
| 4 | 2 | 3 | 3 | 1 | 1 |
| 5 | 2 | 3 | 3 | 1 | 0 |
| 6 | 2 | 3 | 1 | 1 | 0 |
| 7 | 1 | 3 | 3 | 2 | 0 |

Table A.15: Result of the partitioning of the data in Table A.14 using equal-width bins of size 0.5.

## Appendix A. Example

This section provides an example to illustrate the main concepts and algorithms used in this work.

Table A.14 shows a data set with four continuous attributes $(a, b, c, d)$ ranging from 0 to 1.5 and label ($l$) either 0 or 1. The intuition is that this data set has 7 samples, 4 attributes, and 2 labels. This will be used to demonstrate the concepts in this work. Note that due to only having 2 labels in practice this data set supports only binary classification, while this work considers multi-class classification with unbounded numbers of classes.

Many algorithms and concepts in the literature rely upon the data set being based on discrete rather than continuous values. In practice, data sets are binned to support this. For the data set presented in Table A.14, the attributes can be partitioned using equal-width bins of size 0.5, as presented in Table A.15.

Remark that when dealing with a non-binary data set, each *item* is usually given by an attribute-value pair. Thus, each attribute-value pair of Table A.15 is an *item* in our example, such as $(a, 1)$. Table A.16 shows the set of tids, the set of items and the binary relation between them derived from Table A.15. Observe that each column represents an attribute-value pair and an $\times$ indicates in this row this attribute has this value.

Table A.17 shows the frequent itemsets from the data set exhibited in Table A.16 for $minsup = 2$. Note that, for instance, $\mathbf{t}(\{(c, 3)\}) = \{4, 5, 7\}$ and $\mathbf{t}(\{(c, 3), (d, 1)\}) = \{4, 5\}$. Therefore, $sup(\{(c, 3)\}) = 3$ and $sup(\{(c, 3), (d, 1)\}) = 2$.

The CFIs from Table A.17 are marked with an $\times$ in the column 'CFI'. Note that we have 29 frequent

| | $(a,1)$ | $(a,2)$ | $(a,3)$ | $(b,1)$ | $(b,2)$ | $(b,3)$ | $(c,1)$ | $(c,2)$ | $(c,3)$ | $(d,1)$ | $(d,2)$ | $(l,0)$ | $(l,1)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | × | × | | | × | | | × | | | × |
| 2 | | | × | × | | | × | | | × | | | × |
| 3 | | | × | | × | | | × | | × | | | × |
| 4 | | × | | | | × | | | × | × | | | × |
| 5 | | × | | | | × | | | × | × | | × | |
| 6 | | × | | | | × | × | | | × | | × | |
| 7 | × | | | | | × | | | × | | × | × | |

Table A.16: Data of Table A.15 presented using a binary relation between its columns and rows.

| # | **Itemset** | **sup** | **Equiv** | **MG** | **CFI** |
|---|---|---|---|---|---|
| 1 | $\{(a,2)\}$ | 3 | 1 | × | |
| 2 | $\{(a,3)\}$ | 3 | 2 | × | |
| 3 | $\{(b,1)\}$ | 2 | 3 | × | |
| 4 | $\{(b,3)\}$ | 4 | 4 | × | × |
| 5 | $\{(c,1)\}$ | 3 | 5 | × | |
| 6 | $\{(c,3)\}$ | 3 | 6 | × | |
| 7 | $\{(d,1)\}$ | 6 | 7 | × | × |
| 8 | $\{(a,2),(b,3)\}$ | 3 | 1 | | |
| 9 | $\{(a,2),(c,3)\}$ | 2 | 8 | × | |
| 10 | $\{(a,2),(d,1)\}$ | 3 | 1 | | |
| 11 | $\{(a,3),(b,1)\}$ | 2 | 3 | | |
| 12 | $\{(a,3),(c,1)\}$ | 2 | 3 | × | |
| 13 | $\{(a,3),(d,1)\}$ | 3 | 2 | | × |
| 14 | $\{(b,1),(c,1)\}$ | 2 | 3 | | |
| 15 | $\{(b,1),(d,1)\}$ | 2 | 3 | | |
| 16 | $\{(b,3),(c,3)\}$ | 3 | 6 | | × |
| 17 | $\{(b,3),(d,1)\}$ | 3 | 1 | × | |
| 18 | $\{(c,1),(d,1)\}$ | 3 | 5 | | × |
| 19 | $\{(c,3),(d,1)\}$ | 2 | 8 | × | |
| 20 | $\{(a,2),(b,3),(c,3)\}$ | 2 | 8 | | |
| 21 | $\{(a,2),(b,3),(d,1)\}$ | 3 | 1 | | × |
| 22 | $\{(a,2),(c,3),(d,1)\}$ | 2 | 8 | | |
| 23 | $\{(a,3),(b,1),(c,1)\}$ | 2 | 3 | | |
| 24 | $\{(a,3),(b,1),(d,1)\}$ | 2 | 3 | | |
| 25 | $\{(a,3),(c,1),(d,1)\}$ | 2 | 3 | | |
| 26 | $\{(b,1),(c,1),(d,1),\}$ | 2 | 3 | | |
| 27 | $\{(b,3),(c,3),(d,1)\}$ | 2 | 8 | | |
| 28 | $\{(a,2),(b,3),(c,3),(d,1)\}$ | 2 | 8 | | × |
| 29 | $\{(a,3),(b,1),(c,1),(d,1)\}$ | 2 | 3 | | × |

Table A.17: Frequent itemsets mined from the data set exhibited in Table A.16 using $minsup = 2$.

itemsets, while we have only 8 CFIs. The column 'Equiv' of Table A.17 indicates the equivalence class of each of the 29 frequent itemsets. The MGs are marked with an × in the column 'MG'. We have 8 different equivalence classes, with 2 being trivial from these.

715 Table A.18 shows the CARs mined from the data set exhibited in Table A.16 using $minsup = 2$ and $minconf = 60\%$. The number (#) of each CAR is linked to the corresponding itemset in Table A.17. The rules are grouped according to their equivalence class. For convenience, the curly brackets are omitted in the presentation of the rules. For instance, the itemset $\{(d,1)\}$ has support = 6, i.e., it

| # | CAR | sup | conf(%) | comp(%) | princ(%) |
|---|---|---|---|---|---|
| 1 | $(a,2) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 8 | $(a,2),(b,3) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 10 | $(a,2),(d,1) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 17 | $(b,3),(d,1) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 21 | $(a,2),(b,3),(d,1) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 2 | $(a,3) \to 1$ | 3 | 100.00 | 75.00 | 95.00 |
| 13 | $(a,3),(d,1) \to 1$ | 3 | 100.00 | 75.00 | 95.00 |
| 3 | $(b,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 11 | $(a,3),(b,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 12 | $(a,3),(c,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 14 | $(b,1),(c,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 15 | $(b,1),(d,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 23 | $(a,3),(b,1),(c,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 24 | $(a,3),(b,1),(d,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 25 | $(a,3),(c,1),(d,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 26 | $(b,1),(c,1),(d,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 29 | $(a,3),(b,1),(c,1),(d,1) \to 1$ | 2 | 100.00 | 50.00 | 90.00 |
| 4 | $(b,3) \to 0$ | 3 | 75.00 | 100.00 | 80.00 |
| 5 | $(c,1) \to 1$ | 2 | 66.67 | 50.00 | 63.33 |
| 18 | $(c,1),(d,1) \to 1$ | 2 | 66.67 | 50.00 | 63.33 |
| 6 | $(c,3) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 16 | $(b,3),(c,3) \to 0$ | 2 | 66.67 | 66.67 | 66.67 |
| 7 | $(d,1) \to 1$ | 4 | 66.67 | 100.00 | 73.33 |

Table A.18: CARs mined from the data set exhibited in Table A.16 using $minsup = 2$ and $minconf = 60\%$. The principality metric was computed using $\lambda = 0.8$.

appears in 6 from the 7 transactions of Table A.16. From these 6 transactions, 4 transactions belong to class label 1. So, the rule $R : (d,1) \to 1$ has support = 4, confidence = 66.67% and completeness = 100%.

The itemset $\{(a,3),(b,1),(c,1),(d,1)\}$ is closed. Its equivalence class and MGs are presented in Table A.17. This equivalence class has 10 itemsets. As we have two class labels in our example, there are two possible rules for each condition. So, we can write 20 different rules with the itemsets in this equivalence class. The rules have support = 2, confidence = 100% and completeness = 50% for class label 1 in the consequent part of the rule (as we can see in Table A.18), the rules have support = 0, confidence = 0%, and completeness = 0% for class label 0 in the consequent part of the rule. Therefore, the best rules that we can write with the itemsets in this equivalence class is for class label 1. Further, among the best rules, $(a,3),(b,1),(c,1),(d,1) \to 1$ is the most specific rule, while $(b,1) \to 1$ and $(a,3),(c,1) \to 1$ are the most general rules.

*Appendix A.1. Classification Based on Associations (CBA)*

Note that $(a,3) \to 1$ is the best rule according to the CBA's ranking (see Table A.18). This rule covers 3 from the 4 samples of class label 1. As there is not any other rule that can cover the remaining sample of class label 1, this is the only rule selected for class label 1. Considering class label 0 in the

default rule, this would be the only rule selected to the classifier, yielding:

$$\begin{aligned} (a,3) &\rightarrow 1 \quad , \\ \emptyset &\rightarrow 0 \quad . \end{aligned}$$

We are using $\emptyset$ to indicate the default rule. Thus, this classifier could be read as:
*if* $(a,3)$ *then* $(l,1)$ *else* $(l,0)$. Note that $\{(a,3)\}$ is a minimal generator of minimum size.

*Appendix A.2. Classification Based on Multiple Association Rules (CMAR)*

Considering $\gamma = 3.8415$ and $\delta = 3$, CMAR would selected the following three rules from Table A.18 to create the classifier:

$$\begin{aligned} (a,3) &\rightarrow 1 \quad , \\ (a,3),(d,1) &\rightarrow 1 \quad , \\ (b,3) &\rightarrow 0 \quad . \end{aligned}$$

Observe that this differs from CBA in the choice of rules, since CBA (a) selects only rules whose conditions are minimal generators, and (b) CBA attempts to cover a sample only once.

*Appendix A.3. Principal Association Mining (PAM)*

To exemplify the label-based pruning of PAM, let us consider the rules $R_1 : (d,1) \rightarrow 1$ and $R_2 : (a,3),(d,1) \rightarrow 1$. $R_2$ is a more specific rule than $R_1$, and $conf(R_2) = 100\% > conf(R_1) = 66.67\%$, so $R_1$ is pruned.

Letting $\lambda = 0.8$, note that $(a,3),(d,1) \rightarrow 1$ is the best rule according to the PAM's criteria (see Table A.18). For the same reasons as in CBA and considering class label 0 in the default rule, this would be the only rule selected to the classifier, yielding:

$$\begin{aligned} (a,3),(d,1) &\rightarrow 1 \quad , \\ \emptyset &\rightarrow 0 \quad . \end{aligned}$$

The resulting classifiers of CBA and PAM are very similar in this example. The difference between them is that CBA selected a general version of the rule selected by PAM.

*Appendix A.4. Best Rule (BR)*

As our example data set has only two class labels, BR classifier must have two rules. As one of them can be the default rule, we need to select just one rule. Thus, in this example, BR and PAM would provide the same classifier.

| # | CAR | sup | conf(%) | comp(%) |
|---|---|---|---|---|
| $R_{1o}$ | $a[0.3, 0.6], b[1.4, 1.4], d[0.3, 0.7] \rightarrow 0$ | 2 | 100.00 | 66.67 |
| $R_{2o}$ | $a[0.6, 0.9], b[1.2, 1.4], d[0.2, 0.3] \rightarrow 0$ | 2 | 66.67 | 66.67 |
| $R_{3o}$ | $a[0.9, 1.4], b[0.9, 1.3], d[0.2, 0.4] \rightarrow 1$ | 2 | 66.67 | 50.00 |
| $R_{4o}$ | $a[1.4, 1.5], d[0.3, 0.4] \rightarrow 1$ | 3 | 100.00 | 75.00 |
| $R_{5o}$ | $a[1.5, 1.5], b[0.0, 0.2], c[0.1, 0.4], d[0.3, 0.4] \rightarrow 1$ | 2 | 100.00 | 50.00 |
| $R_{6o}$ | $a[1.4, 1.5], c[0.4, 0.8], d[0.4, 0.4] \rightarrow 1$ | 2 | 100.00 | 50.00 |
| $R_{7o}$ | $b[0.9, 1.4], d[0.2, 0.7] \rightarrow 0$ | 3 | 60.00 | 100.00 |
| $R_{8o}$ | $b[1.2, 1.4], c[1.3, 1.4], d[0.2, 0.7] \rightarrow 0$ | 2 | 66.67 | 66.67 |
| $R_{9o}$ | $c[0.1, 0.4], d[0.3, 0.4] \rightarrow 1$ | 2 | 66.67 | 50.00 |

Table A.19: CARs mined from the data set exhibited in Table A.14 using $minsup = 2$, $minconf = 60\%$ and online partitioning with bin size = 0.5. Rules' conditions are closed frequent itemsets. For convenience, we are omitting the symbol $\in$ in the intervals.

*Appendix A.5. A priori versus online partitioning*

Let us compare the mining results of CARs, using a priori and online partitioning, from our example data set. The parameters used in the pattern mining were $minsup = 2$ and $minconf = 60\%$. The binning size for the online partitioning was the same as that used for the a priori partitioning, which is 0.5. Table A.18 and Table A.19 show the results for a priori and online partitioning, respectively. For convenience, **the comparison is restricted to rules whose conditions are CFIs**.

It is know that a priori partitioning leads to incomplete solutions when compared to online partitioning [54]. This explains why there are rules in Table A.19 that do not have a correspondent rule in Table A.18, and vice versa. Rules $R_{1o}$, $R_{3o}$ and $R_{6o}$ from Table A.19 do not have a correspondent rule in Table A.18. Note that they have attributes in intervals that can not be detected due to the a priori partitioning, such as $a \in [0.3, 0.6]$ in rule $R_{1o}$. Rule $R_7$ from Table A.18 does not have a correspondent rule in Table A.19. It happens because the online partitioning would find the itemset $\{d[0.2, 0.7]\}$ and the rule $d[0.2, 0.7] \rightarrow 1$ has confidence = 57.14%, which is less than the user-defined parameter $minconf$.

The correspondent rules are: $R_{21} \equiv R_{2o}$, $R_{13} \equiv R_{4o}$, $R_{29} \equiv R_{5o}$, $R_{16} \equiv R_{8o}$, and $R_{18} \equiv R_{9o}$. Note that the rules with online partitioning tend to be more specific than the rules with a priori partitioning. For instance, the rule $R_{13}$ could be rewrite as $a(1.00, 1.50], d[0.00, 0.50] \rightarrow 1$, being more general than the rule $R_{4o}$. Moreover, rule $R_{8o}$ has also an attribute more than rule $R_{16}$. Rules $R_4$ from Table A.18 and $R_{7o}$ from Table A.19 have also some correspondence, but they have different confidences.

## Appendix B. Features

The full list of features can be found in Table B.20.

| ID | Type | Description |
|---|---|---|
| 1 | *Boolean* | DLL can be relocated at load time. *Extracted from DLLs characteristics* |

| | | |
|---|---|---|
| 2 | *Boolean* | Code Integrity checks are enforced. *Extracted from DLLs characteristics* |
| 3 | *Boolean* | Image is NX compatible. *Extracted from DLLs characteristics* |
| 4 | *Boolean* | Isolation aware, but do not isolate the image. *Extracted from DLLs characteristics* |
| 5 | *Boolean* | Does not use structured exception (SE) handling. No SE handler may be called in this image. *Extracted from DLLs characteristics* |
| 6 | *Boolean* | Do not bind the image. *Extracted from DLLs characteristics* |
| 7 | *Boolean* | A WDM driver. *Extracted from DLLs characteristics* |
| 8 | *Boolean* | Terminal Server aware. *Extracted from DLLs characteristics* |
| 9 | *Integer* | The image file checksum. |
| 10 | *Integer* | The preferred address of the first byte of image when loaded into memory |
| 11 | *Integer* | The address that is relative to the image base of the beginning-of-code section when it is loaded into memory. |
| 12 | *Integer* | The major version number of the required operating system. |
| 13 | *Integer* | The minor version number of the required operating system. |
| 14 | *Integer* | The size (in bytes) of the image, including all headers, as the image is loaded in memory. |
| 15 | *Integer* | The size of the code (.text) section, or the sum of all code sections if there are multiple sections. |
| 16 | *Integer* | The combined size of an MS DOS stub, PE header, and section headers rounded up to a multiple of FileAlignment. |
| 17 | *Integer* | The size of the initialized data section, or the sum of all such sections if there are multiple data sections. |
| 18 | *Integer* | The size of the uninitialized data section (BSS), or the sum of all such sections if there are multiple BSS sections |
| 19 | *Integer* | The size of the stack to reserve. |
| 20 | *Integer* | The size of the stack to commit. |
| 21 | *Integer* | The alignment (in bytes) of sections when they are loaded into memory. |
| 22 | *Integer* | The number of standard sections the PE holds |
| 23 | *Integer* | The number of non-standard sections the PE holds |
| 24 | *Float* | The ratio between the number of standard sections found and the number of all sections found in the PE under analysis |
| 25 | *Integer* | The number of Executable sections the PE holds |

| | | |
|---|---|---|
| 26 | *Integer* | The number of Writable sections the PE holds |
| 27 | *Integer* | The number of Writable and Executable sections the PE holds |
| 28 | *Integer* | The number of readable and executable sections |
| 29 | *Integer* | The number of readable and writable sections |
| 30 | *Integer* | The number of Writable and Readable and Executable sections the PE holds |
| 31 | *Boolean* | The code section is not executable |
| 32 | *Boolean* | The executable section is not a code section |
| 33 | *Boolean* | The code section is not present in the PE under analysis |
| 34 | *Boolean* | The entry point is not in the code section |
| 35 | *Boolean* | The entry point is not in a standard section |
| 36 | *Boolean* | The entry point is not in an executable section |
| 37 | *Float* | The ratio between raw data and virtual size for the section of the entry point |
| 38 | *Integer* | The number of section having their raw data size zero |
| 39 | *Integer* | The number of sections having their virtual size greater than their raw data size. |
| 40 | *Float* | The maximum ratio raw data to virtual size among all sections |
| 41 | *Float* | the minimum ratio raw data to virtual size among all sections |
| 42 | *Boolean* | The address pointing to raw data on disk is not conforming with the file alignement |
| 43 | *Float* $\in [0; 8]$ | The byte entropy of code (.text) sections |
| 44 | *Float* $\in [0; 8]$ | The byte entropy of data section |
| 45 | *Float* $\in [0; 8]$ | The byte entropy of resource section |
| 46 | *Float* $\in [0; 8]$ | The byte entropy of PE header |
| 47 | *Float* $\in [0; 8]$ | The byte entropy of the entire PE file |
| 48 | *Float* $\in [0; 8]$ | The byte entropy of the section holding the entry point of the PE under analysis |
| 49 - 112 | *Integer* $\in [0; 255]$ | Values of 64 bytes following the entry point, each byte correspond to 1 feature position |
| 113 | *Integer* | The number of DLLs imported |
| 114 | *Integer* | The number of functions imported found in the import table directory (IDT) |

| | | |
|---|---|---|
| 115 | *Integer* | The number of malicious APIs imported (malicious as defined in [57]). This list includes: "GetProcAddress", "LoadLibraryA", "LoadLibrary", "ExitProcess", "GetModuleHandleA", "VirtualAlloc", "VirtualFree", "GetModuleFileNameA", "CreateFileA", "RegQueryValueExA", "MessageBoxA", "GetCommandLineA", "VirtualProtect", "GetStartupInfoA", "GetStdHandle", "RegOpenKeyExA". |
| 116 | *Float* | The ratio between the number of malicious APIs imported to the number of all functions imported by the PE |
| 117 | *Integer* | the number of addresses (corresponds to functions) found in the import address table (IAT) |
| 118 | *Boolean* | The debug directory is present or not |
| 119 | *Integer* | The number of resources the PE holds |

Table B.20: Full list of all features descriptions.

## Appendix C. Classifiers

### *Appendix C.1. BR*

**if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_8 = 0$ **and** $f_9 = 4128768$ **and** $f_{10} = 4096$ **and** $f_{11} = 4$ **and** $f_{12} = 1$ **and** $f_{13} = 6115328$ **and** $f_{14} = 8192$ **and** $f_{15} = 4096$ **and** $f_{16} = 126464$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{21} = 1$ **and** $f_{22} = 2$ **and** $f_{23} = 1$ **and** $f_{24} = 1$ **and** $f_{25} = 1$ **and** $f_{26} = 1$ **and** $f_{27} = 1$ **and** $f_{28} = 2$ **and** $f_{29} = 1$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{36} = 1$ **and** $f_{37} = 0$ **and** $f_{38} = 2$ **and** $f_{39} = 1$ **and** $f_{40} = 1$ **and** $f_{41} = 0$ **and** $f_{45} = 10$ **and** $f_{46} = 1$ **and** $f_{47} = 16$ **and** $f_{48} = 189$ **and** $f_{49} = 44$ **and** $f_{50} = 78$ **and** $f_{51} = 66$ **and** $f_{52} = 0$ **and** $f_{53} = 199$ **and** $f_{54} = 69$ **and** $f_{55} = 0$ **and** $f_{56} = 212$ **and** $f_{57} = 0$ **and** $f_{58} = 63$ **and** $f_{59} = 0$ **and** $f_{60} = 184$ **and** $f_{61} = 212$ **and** $f_{62} = 235$ **and** $f_{63} = 63$ **and** $f_{64} = 0$ **and** $f_{65} = 137$ **and** $f_{66} = 69$ **and** $f_{67} = 4$ **and** $f_{68} = 137$ **and** $f_{69} = 69$ **and** $f_{70} = 84$ **and** $f_{71} = 80$ **and** $f_{72} = 199$ **and** $f_{73} = 69$ **and** $f_{74} = 16$ **and** $f_{75} = 178$ **and** $f_{76} = 59$ **and** $f_{77} = 2$ **and** $f_{78} = 0$ **and** $f_{79} = 255$ **and** $f_{80} = 77$ **and** $f_{81} = 12$ **and** $f_{82} = 255$ **and** $f_{83} = 69$ **and** $f_{84} = 20$ **and** $f_{85} = 255$ **and** $f_{86} = 69$ **and** $f_{87} = 88$ **and** $f_{88} = 198$ **and** $f_{89} = 69$ **and** $f_{90} = 28$ **and** $f_{91} = 8$ **and** $f_{92} = 184$ **and** $f_{93} = 0$ **and** $f_{94} = 8$ **and** $f_{95} = 0$ **and** $f_{96} = 0$ **and** $f_{97} = 141$ **and** $f_{98} = 125$ **and** $f_{99} = 48$ **and** $f_{100} = 171$ **and** $f_{101} = 171$ **and** $f_{102} = 171$ **and** $f_{103} = 171$ **and** $f_{104} = 187$ **and** $f_{105} = 0$ **and** $f_{106} = 0$ **and** $f_{107} = 216$ **and** $f_{108} = 0$ **and** $f_{109} = 191$ **and** $f_{110} = 120$ **and** $f_{111} = 87$ **and** $f_{112} = 1$ **and** $f_{113} = 2$ **and** $f_{114} = 2$ **and** $f_{115} = 11$ **and** $f_{117} = 1$ **then** $class$ =kkrunchy

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{12} = 1$ **and** $f_{14} = 0$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{24} = 0$ **and** $f_{26} = 0$ **and** $f_{27} = 0$ **and** $f_{29} = 0$ **and** $f_{31} = 0$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 1$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 233$ **and** $f_{49} = 37$ **and** $f_{50} = 228$ **and** $f_{51} = 255$ **and** $f_{52} = 255$ **and** $f_{53} = 0$ **and** $f_{54} = 0$ **and** $f_{55} = 0$ **and** $f_{60} = 30$ **and** $f_{63} = 0$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 0$ **and** $f_{69} = 0$ **and** $f_{70} = 0$ **and** $f_{71} = 0$ **and** $f_{72} = 62$ **and** $f_{75} = 0$ **and** $f_{76} = 46$ **and** $f_{79} = 0$ **and** $f_{80} = 38$ **and** $f_{83} = 0$ **and** $f_{84} = 0$ **and** $f_{85} = 0$ **and** $f_{86} = 0$ **and** $f_{87} = 0$ **and** $f_{88} = 0$ **and** $f_{89} = 0$ **and** $f_{90} = 0$ **and** $f_{91} = 0$ **and** $f_{92} = 75$ **and** $f_{95} = 0$ **and** $f_{96} = 54$ **and** $f_{99} = 0$ **and** $f_{100} = 0$ **and** $f_{101} = 0$ **and** $f_{102} = 0$ **and** $f_{103} = 0$ **and** $f_{104} = 0$ **and** $f_{105} = 0$ **and** $f_{106} = 0$ **and** $f_{107} = 0$ **and** $f_{108} = 0$ **and** $f_{109} = 0$ **and** $f_{110} = 0$ **and** $f_{111} = 0$ **and** $f_{112} = 2$ **and** $f_{113} = 2$ **and** $f_{114} = 2$ **and** $f_{115} = 11$ **and** $f_{117} = 1$ **then** $class$ =telock

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_8 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} = 1$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{24} = 1$ **and** $f_{26} = 1$ **and** $f_{27} = 1$ **and** $f_{29} = 1$ **and** $f_{31} = 1$ **and** $f_{33} = 1$ **and** $f_{34} =$

$1$ **and** $f_{35} = 0$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{47} = 13$ **and** $f_{48} = 233$ **and** $f_{49} = 166$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{56} = 0$ **and** $f_{60} = 0$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 0$ **and** $f_{71} = 0$ **and** $f_{72} = 0$ **and** $f_{76} = 0$ **and** $f_{77} = 78$ **and** $f_{78} = 101$ **and** $f_{79} = 111$ **and** $f_{80} = 76$ **and** $f_{81} = 105$ **and** $f_{82} = 116$ **and** $f_{83} = 101$ **and** $f_{84} = 32$ **and** $f_{85} = 69$ **and** $f_{86} = 120$ **and** $f_{87} = 101$ **and** $f_{88} = 99$ **and** $f_{89} = 117$ **and** $f_{90} = 116$ **and** $f_{91} = 97$ **and** $f_{92} = 98$ **and** $f_{93} = 108$ **and** $f_{94} = 101$ **and** $f_{95} = 32$ **and** $f_{96} = 70$ **and** $f_{97} = 105$ **and** $f_{98} = 108$ **and** $f_{99} = 101$ **and** $f_{100} = 32$ **and** $f_{101} = 67$ **and** $f_{102} = 111$ **and** $f_{103} = 109$ **and** $f_{104} = 112$ **and** $f_{105} = 114$ **and** $f_{106} = 101$ **and** $f_{107} = 115$ **and** $f_{108} = 115$ **and** $f_{109} = 111$ **and** $f_{110} = 114$ **and** $f_{111} = 13$ **and** $f_{117} = 1$ **then** $class$ =neolite

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} = 1$ **and** $f_{20} = 4096$ **and** $f_{21} = 1$ **and** $f_{24} = 2$ **and** $f_{26} = 2$ **and** $f_{27} = 2$ **and** $f_{29} = 2$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{37} = 0$ **and** $f_{38} = 2$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 232$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{53} = 0$ **and** $f_{54} = 88$ **and** $f_{55} = 5$ **and** $f_{58} = 0$ **and** $f_{59} = 0$ **and** $f_{60} = 139$ **and** $f_{61} = 48$ **and** $f_{62} = 3$ **and** $f_{63} = 240$ **and** $f_{64} = 43$ **and** $f_{65} = 192$ **and** $f_{66} = 139$ **and** $f_{67} = 254$ **and** $f_{68} = 102$ **and** $f_{69} = 173$ **and** $f_{70} = 193$ **and** $f_{71} = 224$ **and** $f_{72} = 12$ **and** $f_{73} = 139$ **and** $f_{74} = 200$ **and** $f_{75} = 80$ **and** $f_{76} = 173$ **and** $f_{77} = 43$ **and** $f_{78} = 200$ **and** $f_{79} = 3$ **and** $f_{80} = 241$ **and** $f_{81} = 139$ **and** $f_{82} = 200$ **and** $f_{83} = 87$ **and** $f_{84} = 81$ **and** $f_{85} = 73$ **and** $f_{86} = 138$ **and** $f_{87} = 68$ **and** $f_{88} = 57$ **and** $f_{89} = 6$ **and** $f_{90} = 136$ **and** $f_{91} = 4$ **and** $f_{92} = 49$ **and** $f_{93} = 117$ **and** $f_{94} = 246$ **and** $f_{114} = 2$ **and** $f_{116} = 2$ **and** $f_{117} = 1$ **then** $class$ =mpress

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{11} = 5$ **and** $f_{12} = 1$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 232$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{53} = 0$ **and** $f_{54} = 91$ **and** $f_{55} = 141$ **and** $f_{56} = 91$ **and** $f_{57} = 198$ **and** $f_{58} = 1$ **and** $f_{59} = 27$ **and** $f_{60} = 139$ **and** $f_{61} = 19$ **and** $f_{62} = 141$ **and** $f_{63} = 115$ **and** $f_{64} = 20$ **and** $f_{65} = 106$ **and** $f_{66} = 8$ **and** $f_{67} = 89$ **and** $f_{68} = 1$ **and** $f_{69} = 22$ **and** $f_{70} = 173$ **and** $f_{71} = 73$ **and** $f_{72} = 117$ **and** $f_{73} = 250$ **and** $f_{74} = 139$ **and** $f_{75} = 232$ **and** $f_{76} = 198$ **and** $f_{77} = 6$ **and** $f_{78} = 233$ **and** $f_{79} = 139$ **and** $f_{80} = 67$ **and** $f_{81} = 12$ **and** $f_{82} = 137$ **and** $f_{83} = 70$ **and** $f_{84} = 1$ **and** $f_{114} = 3$ **and** $f_{117} = 1$ **then** $class$ =pacman

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} = 1$ **and** $f_{20} = 4096$ **and** $f_{24} = 2$ **and** $f_{26} = 2$ **and** $f_{27} = 2$ **and** $f_{29} = 2$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 184$ **and** $f_{52} = 0$ **and** $f_{53} = 80$ **and** $f_{54} = 100$ **and** $f_{55} = 255$ **and** $f_{56} = 53$ **and** $f_{57} = 0$ **and** $f_{58} = 0$ **and** $f_{59} = 0$ **and** $f_{60} = 0$ **and** $f_{61} = 100$ **and** $f_{62} = 137$ **and** $f_{63} = 37$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 51$ **and** $f_{69} = 192$ **and** $f_{70} = 137$ **and** $f_{71} = 8$ **and** $f_{72} = 80$ **and** $f_{73} = 69$ **and** $f_{74} = 67$ **and** $f_{75} = 111$ **and** $f_{76} = 109$ **and** $f_{77} = 112$ **and** $f_{78} = 97$ **and** $f_{79} = 99$ **and** $f_{80} = 116$ **and** $f_{81} = 50$ **and** $f_{82} = 0$ **and** $f_{114} = 3$ **then** $class$ =pecompact

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{11} = 4$ **and** $f_{12} = 1$ **and** $f_{20} = 4096$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{40} = 1$ **and** $f_{41} = 0$ **and** $f_{47} = 16$ **and** $f_{48} = 85$ **and** $f_{49} = 139$ **and** $f_{50} = 236$ **and** $f_{51} = 131$ **and** $f_{52} = 196$ **and** $f_{53} = 240$ **and** $f_{54} = 184$ **and** $f_{55} = 0$ **and** $f_{56} = 16$ **and** $f_{57} = 64$ **and** $f_{58} = 0$ **and** $f_{59} = 232$ **and** $f_{60} = 1$ **and** $f_{61} = 0$ **and** $f_{62} = 0$ **and** $f_{63} = 0$ **and** $f_{64} = 154$ **and** $f_{65} = 131$ **and** $f_{66} = 196$ **and** $f_{67} = 16$ **and** $f_{68} = 139$ **and** $f_{69} = 229$ **and** $f_{70} = 93$ **and** $f_{71} = 233$ **and** $f_{75} = 0$ **and** $f_{117} = 1$ **then** $class$ =enigma

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} = 1$ **and** $f_{20} = 4096$ **and** $f_{21} = 1$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{37} = 1$ **and** $f_{39} = 1$ **and** $f_{40} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 190$ **and** $f_{53} = 0$ **and** $f_{54} = 141$ **and** $f_{55} = 190$ **and** $f_{59} = 255$ **and** $f_{117} = 1$ **then** $class$ =upx

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} = 1$ **and** $f_{20} = 4096$ **and** $f_{41} = 0$ **and** $f_{48} = 104$ **and** $f_{53} = 232$ **then** $class$ =vmprotect

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{10} = 4096$ **and** $f_{12} = 1$ **and** $f_{17} = 0$ **and** $f_{20} = 4096$ **and** $f_{22} = 1$ **and** $f_{23} = 11$ **and** $f_{26} = 0$ **and** $f_{29} = 0$ **and** $f_{31} = 0$ **and** $f_{32} = 0$ **and** $f_{33} = 0$ **and** $f_{34} = 0$ **and** $f_{37} = 0$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{53} = 186$ **and** $f_{57} = 0$ **then** $class$ =stealth

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{12} = 1$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{24} = 0$ **and** $f_{26} = 0$ **and** $f_{27} = 0$ **and** $f_{29} = 0$ **and** $f_{31} = 0$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 1$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{52} = 0$ **and** $f_{53} = 0$ **and** $f_{117} = 1$ **then** $class$ =aspack

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} = 1$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{21} = 3$ **and** $f_{26} = 0$ **and** $f_{29} = 0$ **and** $f_{30} = 0$ **and** $f_{32} = 0$ **and** $f_{35} = 0$ **and** $f_{39} = 1$ **and** $f_{41} = 0$ **and** $f_{68} = 0$ **and** $f_{73} = 0$ **and** $f_{91} = 0$ **and** $f_{101} = 0$ **and** $f_{115} = 1$ **then** $class$ =armadillo

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_8 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{11} = 4$ **and** $f_{12} = 1$ **and** $f_{14} = 13824$ **and** $f_{15} = 1024$ **and** $f_{16} = 43008$ **and** $f_{17} = 65536$ **and** $f_{18} = 1048576$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{21} = 2$ **and** $f_{30} = 0$ **and** $f_{32} =$

$0$ **and** $f_{33} = 0$ **and** $f_{34} = 0$ **and** $f_{35} = 0$ **and** $f_{36} = 11$ **and** $f_{37} = 1$ **and** $f_{39} = 1$ **and** $f_{40} = 1$ **and** $f_{41} = 0$ **and** $f_{43} = 32$ **and** $f_{112} = 1$ **and** $f_{115} = 1$ **and** $f_{117} = 1$ **then** $class$ =nspack

**else** $class$ =not packed

*Appendix C.2. BR-R*

**if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_8 = 0$ **and** $f_9 = 4128768$ **and** $f_{10} = 4096$ **and** $f_{11} = 4$ **and** $f_{12} = 0$ **and** $f_{13} = 6115328$ **and** $f_{14} = 8192$ **and** $f_{15} = 4096$ **and** $f_{16} = 126464$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{21} = 0$ **and** $f_{22} = 1$ **and** $f_{23} = 0$ **and** $f_{24} = 1$ **and** $f_{25} = 1$ **and** $f_{26} = 1$ **and** $f_{27} = 1$ **and** $f_{28} = 1$ **and** $f_{29} = 1$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{36} = 0$ **and** $f_{37} = 0$ **and** $f_{38} = 1$ **and** $f_{39} = 0$ **and** $f_{40} = 0$ **and** $f_{41} = 0$ **and** $f_{45} = 2$ **and** $f_{46} \in [0.1, 0.1]$ **and** $f_{47} = 7$ **and** $f_{48} = 189$ **and** $f_{49} = 44$ **and** $f_{50} = 78$ **and** $f_{51} = 66$ **and** $f_{52} = 0$ **and** $f_{53} = 199$ **and** $f_{54} = 69$ **and** $f_{55} = 0$ **and** $f_{56} = 212$ **and** $f_{57} = 0$ **and** $f_{58} = 63$ **and** $f_{59} = 0$ **and** $f_{60} = 184$ **and** $f_{61} = 212$ **and** $f_{62} = 235$ **and** $f_{63} = 63$ **and** $f_{64} = 0$ **and** $f_{65} = 137$ **and** $f_{66} = 69$ **and** $f_{67} = 4$ **and** $f_{68} = 137$ **and** $f_{69} = 69$ **and** $f_{70} = 84$ **and** $f_{71} = 80$ **and** $f_{72} = 199$ **and** $f_{73} = 69$ **and** $f_{74} = 16$ **and** $f_{75} = 178$ **and** $f_{76} = 59$ **and** $f_{77} = 2$ **and** $f_{78} = 0$ **and** $f_{79} = 255$ **and** $f_{80} = 77$ **and** $f_{81} = 12$ **and** $f_{82} = 255$ **and** $f_{83} = 69$ **and** $f_{84} = 20$ **and** $f_{85} = 255$ **and** $f_{86} = 69$ **and** $f_{87} = 88$ **and** $f_{88} = 198$ **and** $f_{89} = 69$ **and** $f_{90} = 28$ **and** $f_{91} = 8$ **and** $f_{92} = 184$ **and** $f_{93} = 0$ **and** $f_{94} = 8$ **and** $f_{95} = 0$ **and** $f_{96} = 0$ **and** $f_{97} = 141$ **and** $f_{98} = 125$ **and** $f_{99} = 48$ **and** $f_{100} = 171$ **and** $f_{101} = 171$ **and** $f_{102} = 171$ **and** $f_{103} = 171$ **and** $f_{104} = 187$ **and** $f_{105} = 0$ **and** $f_{106} = 0$ **and** $f_{107} = 216$ **and** $f_{108} = 0$ **and** $f_{109} = 191$ **and** $f_{110} = 120$ **and** $f_{111} = 87$ **and** $f_{112} = 1$ **and** $f_{113} = 1$ **and** $f_{114} = 2$ **and** $f_{115} = 2$ **and** $f_{117} = 1$ **then** $class$ =kkrunchy

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{12} = 0$ **and** $f_{14} = 0$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{24} = 0$ **and** $f_{26} = 0$ **and** $f_{27} = 0$ **and** $f_{29} = 0$ **and** $f_{31} = 0$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 1$ **and** $f_{39} \in [0.7, 1.2]$ **and** $f_{41} = 0$ **and** $f_{48} = 233$ **and** $f_{49} = 37$ **and** $f_{50} = 228$ **and** $f_{51} = 255$ **and** $f_{52} = 255$ **and** $f_{53} = 0$ **and** $f_{54} = 0$ **and** $f_{55} = 0$ **and** $f_{60} = 30$ **and** $f_{63} = 0$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 0$ **and** $f_{69} = 0$ **and** $f_{70} = 0$ **and** $f_{71} = 0$ **and** $f_{72} = 62$ **and** $f_{75} = 0$ **and** $f_{76} = 46$ **and** $f_{79} = 0$ **and** $f_{80} = 38$ **and** $f_{83} = 0$ **and** $f_{84} = 0$ **and** $f_{85} = 0$ **and** $f_{86} = 0$ **and** $f_{87} = 0$ **and** $f_{88} = 0$ **and** $f_{89} = 0$ **and** $f_{90} = 0$ **and** $f_{91} = 0$ **and** $f_{92} = 75$ **and** $f_{95} = 0$ **and** $f_{96} = 54$ **and** $f_{99} = 0$ **and** $f_{100} = 0$ **and** $f_{101} = 0$ **and** $f_{102} = 0$ **and** $f_{103} = 0$ **and** $f_{104} = 0$ **and** $f_{105} = 0$ **and** $f_{106} = 0$ **and** $f_{107} = 0$ **and** $f_{108} = 0$ **and** $f_{109} = 0$ **and** $f_{110} = 0$ **and** $f_{111} = 0$ **and** $f_{112} = 2$ **and** $f_{113} = 1$ **and** $f_{114} = 2$ **and** $f_{115} = 2$ **and** $f_{117} = 1$ **then** $class$ =telock

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_8 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} \in [0.0, 1.0]$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{21} \in [3.0, 5.0]$ **and** $f_{24} = 1$ **and** $f_{26} = 1$ **and** $f_{27} = 1$ **and** $f_{29} = 1$ **and** $f_{31} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{39} \in [1.0, 2.3]$ **and** $f_{41} = 0$ **and** $f_{47} \in [6.2, 6.5]$ **and** $f_{48} = 233$ **and** $f_{49} = 166$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{56} = 0$ **and** $f_{60} = 0$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 0$ **and** $f_{71} = 0$ **and** $f_{72} = 0$ **and** $f_{76} = 0$ **and** $f_{77} = 78$ **and** $f_{78} = 101$ **and** $f_{79} = 111$ **and** $f_{80} = 76$ **and** $f_{81} = 105$ **and** $f_{82} = 116$ **and** $f_{83} = 101$ **and** $f_{84} = 32$ **and** $f_{85} = 69$ **and** $f_{86} = 120$ **and** $f_{87} = 101$ **and** $f_{88} = 99$ **and** $f_{89} = 117$ **and** $f_{90} = 116$ **and** $f_{91} = 97$ **and** $f_{92} = 98$ **and** $f_{93} = 108$ **and** $f_{94} = 101$ **and** $f_{95} = 32$ **and** $f_{96} = 70$ **and** $f_{97} = 105$ **and** $f_{98} = 108$ **and** $f_{99} = 101$ **and** $f_{100} = 32$ **and** $f_{101} = 67$ **and** $f_{102} = 111$ **and** $f_{103} = 109$ **and** $f_{104} = 112$ **and** $f_{105} = 114$ **and** $f_{106} = 101$ **and** $f_{107} = 115$ **and** $f_{108} = 115$ **and** $f_{109} = 111$ **and** $f_{110} = 114$ **and** $f_{111} = 13$ **and** $f_{114} \in [2.0, 4.0]$ **and** $f_{117} = 1$ **then** $class$ =neolite

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} \in [0.0, 1.0]$ **and** $f_{20} = 4096$ **and** $f_{21} \in [0.0, 1.0]$ **and** $f_{24} = 2$ **and** $f_{26} = 2$ **and** $f_{27} = 2$ **and** $f_{28} \in [2.0, 3.0]$ **and** $f_{29} = 2$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{37} = 0$ **and** $f_{38} = 1$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 232$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{53} = 0$ **and** $f_{54} = 88$ **and** $f_{55} = 5$ **and** $f_{58} = 0$ **and** $f_{59} = 0$ **and** $f_{60} = 139$ **and** $f_{61} = 48$ **and** $f_{62} = 3$ **and** $f_{63} = 240$ **and** $f_{64} = 43$ **and** $f_{65} = 192$ **and** $f_{66} = 139$ **and** $f_{67} = 254$ **and** $f_{68} = 102$ **and** $f_{69} = 173$ **and** $f_{70} = 193$ **and** $f_{71} = 224$ **and** $f_{72} = 12$ **and** $f_{73} = 139$ **and** $f_{74} = 200$ **and** $f_{75} = 80$ **and** $f_{76} = 173$ **and** $f_{77} = 43$ **and** $f_{78} = 200$ **and** $f_{79} = 3$ **and** $f_{80} = 241$ **and** $f_{81} = 139$ **and** $f_{82} = 200$ **and** $f_{83} = 87$ **and** $f_{84} = 81$ **and** $f_{85} = 73$ **and** $f_{86} = 138$ **and** $f_{87} = 68$ **and** $f_{88} = 57$ **and** $f_{89} = 6$ **and** $f_{90} = 136$ **and** $f_{91} = 4$ **and** $f_{92} = 49$ **and** $f_{93} = 117$ **and** $f_{94} = 246$ **and** $f_{114} = 2$ **and** $f_{116} = 2$ **and** $f_{117} = 1$ **then** $class$ =mpress

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{11} = 5$ **and** $f_{12} \in [0.0, 1.0]$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{39} \in [0.5, 1.0]$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 232$ **and** $f_{50} = 0$ **and** $f_{51} = 0$ **and** $f_{52} = 0$ **and** $f_{53} = 0$ **and** $f_{54} = 91$ **and** $f_{55} = 141$ **and** $f_{56} = 91$ **and** $f_{57} = 198$ **and** $f_{58} = 1$ **and** $f_{59} = 27$ **and** $f_{60} = 139$ **and** $f_{61} = 19$ **and** $f_{62} = 141$ **and** $f_{63} = 115$ **and** $f_{64} = 20$ **and** $f_{65} = 106$ **and** $f_{66} = 8$ **and** $f_{67} = 89$ **and** $f_{68} = 1$ **and** $f_{69} = 22$ **and** $f_{70} = 173$ **and** $f_{71} = 73$ **and** $f_{72} = 117$ **and** $f_{73} = 250$ **and** $f_{74} =$

139 **and** $f_{75} = 232$ **and** $f_{76} = 198$ **and** $f_{77} = 6$ **and** $f_{78} = 233$ **and** $f_{79} = 139$ **and** $f_{80} = 67$ **and** $f_{81} = 12$ **and** $f_{82} = 137$ **and** $f_{83} = 70$ **and** $f_{84} = 1$ **and** $f_{114} = 5$ **and** $f_{117} = 1$ **then** *class* =pacman

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} = 0$ **and** $f_{20} = 4096$ **and** $f_{24} = 2$ **and** $f_{26} = 2$ **and** $f_{27} = 2$ **and** $f_{28} \in [2.0, 3.0]$ **and** $f_{29} = 2$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{39} \in [0.7, 1.0]$ **and** $f_{41} = 0$ **and** $f_{48} = 184$ **and** $f_{52} = 0$ **and** $f_{53} = 80$ **and** $f_{54} = 100$ **and** $f_{55} = 255$ **and** $f_{56} = 53$ **and** $f_{57} = 0$ **and** $f_{58} = 0$ **and** $f_{59} = 0$ **and** $f_{60} = 0$ **and** $f_{61} = 100$ **and** $f_{62} = 137$ **and** $f_{63} = 37$ **and** $f_{64} = 0$ **and** $f_{65} = 0$ **and** $f_{66} = 0$ **and** $f_{67} = 0$ **and** $f_{68} = 51$ **and** $f_{69} = 192$ **and** $f_{70} = 137$ **and** $f_{71} = 8$ **and** $f_{72} = 80$ **and** $f_{73} = 69$ **and** $f_{74} = 67$ **and** $f_{75} = 111$ **and** $f_{76} = 109$ **and** $f_{77} = 112$ **and** $f_{78} = 97$ **and** $f_{79} = 99$ **and** $f_{80} = 116$ **and** $f_{81} = 50$ **and** $f_{82} = 0$ **and** $f_{114} \in [4.0, 5.0]$ **then** *class* =pecompact

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{11} = 4$ **and** $f_{12} = 0$ **and** $f_{20} = 4096$ **and** $f_{21} \in [1.0, 2.0]$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{40} = 0$ **and** $f_{41} = 0$ **and** $f_{47} \in [7.9, 8.0]$ **and** $f_{48} = 85$ **and** $f_{49} = 139$ **and** $f_{50} = 236$ **and** $f_{51} = 131$ **and** $f_{52} = 196$ **and** $f_{53} = 240$ **and** $f_{54} = 184$ **and** $f_{55} = 0$ **and** $f_{56} = 16$ **and** $f_{57} = 64$ **and** $f_{58} = 0$ **and** $f_{59} = 232$ **and** $f_{60} = 1$ **and** $f_{61} = 0$ **and** $f_{62} = 0$ **and** $f_{63} = 0$ **and** $f_{64} = 154$ **and** $f_{65} = 131$ **and** $f_{66} = 196$ **and** $f_{67} = 16$ **and** $f_{68} = 139$ **and** $f_{69} = 229$ **and** $f_{70} = 93$ **and** $f_{71} = 233$ **and** $f_{75} = 0$ **and** $f_{117} = 1$ **then** *class* =enigma

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} \in [0.0, 1.0]$ **and** $f_{20} = 4096$ **and** $f_{21} \in [0.0, 1.0]$ **and** $f_{28} \in [2.0, 3.0]$ **and** $f_{30} = 0$ **and** $f_{31} = 1$ **and** $f_{32} = 1$ **and** $f_{33} = 1$ **and** $f_{34} = 1$ **and** $f_{35} = 0$ **and** $f_{37} = 1$ **and** $f_{39} \in [0.6, 1.1]$ **and** $f_{40} = 0$ **and** $f_{41} = 0$ **and** $f_{48} = 96$ **and** $f_{49} = 190$ **and** $f_{53} = 0$ **and** $f_{54} = 141$ **and** $f_{55} = 190$ **and** $f_{59} = 255$ **and** $f_{117} = 1$ **then** *class* =upx

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{12} = 0$ **and** $f_{20} = 4096$ **and** $f_{41} = 0$ **and** $f_{48} = 104$ **and** $f_{53} = 232$ **then** *class* =vmprotect

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{10} = 4096$ **and** $f_{12} \in [0.0, 1.0]$ **and** $f_{17} = 0$ **and** $f_{20} = 4096$ **and** $f_{22} = 0$ **and** $f_{23} = 1$ **and** $f_{26} = 0$ **and** $f_{28} \in [1.0, 2.0]$ **and** $f_{29} = 0$ **and** $f_{31} = 0$ **and** $f_{32} = 0$ **and** $f_{33} = 0$ **and** $f_{34} = 0$ **and** $f_{36} \in [1.0, 1.0]$ **and** $f_{37} = 0$ **and** $f_{39} \in [1.0, 1.0]$ **and** $f_{41} = 0$ **and** $f_{53} = 186$ **and** $f_{57} = 0$ **then** *class* =stealth

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{12} = 0$ **and** $f_{17} = 0$ **and** $f_{18} = 1048576$ **and** $f_{20} = 4096$ **and** $f_{35} = 1$ **and** $f_{39} \in [0.7, 1.0]$ **and** $f_{41} = 0$ **and** $f_{114} \in [3.0, 5.0]$ **and** $f_{117} = 1$ **then** *class* =aspack

**else if** $f_1 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_{12} = 0$ **and** $f_{17} = 0$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{21} \in [4.0, 5.0]$ **and** $f_{30} = 0$ **and** $f_{32} = 0$ **and** $f_{35} = 0$ **and** $f_{39} \in [1.0, 7.8]$ **and** $f_{41} = 0$ **and** $f_{115} \in [-0.0, 0.1]$ **and** $f_{116} = 10$ **then** *class* =armadillo

**else if** $f_0 = 0$ **and** $f_1 = 0$ **and** $f_2 = 0$ **and** $f_3 = 0$ **and** $f_4 = 0$ **and** $f_5 = 0$ **and** $f_6 = 0$ **and** $f_7 = 0$ **and** $f_9 = 4194304$ **and** $f_{10} = 4096$ **and** $f_{12} = 0$ **and** $f_{15} = 1024$ **and** $f_{19} = 4096$ **and** $f_{20} = 4096$ **and** $f_{30} = 0$ **and** $f_{32} = 0$ **and** $f_{33} = 0$ **and** $f_{34} = 0$ **and** $f_{35} = 0$ **and** $f_{36} \in [1.0, 1.1]$ **and** $f_{41} = 0$ **and** $f_{115} \in [0.0, 0.2]$ **and** $f_{117} = 1$ **then** *class* =nspack

**else** *class* =not packed

*Appendix C.3. CBA*

**if** $f_{53} = 232$ **and** $f_{48} = 104$ **then** *class* =vmprotect
**else if** $f_{64} = 154$ **then** *class* =enigma
**else if** $f_{36} = 1$ **then** *class* =kkrunchy
**else if** $f_{65} = 192$ **then** *class* =mpress
**else if** $f_{49} = 166$ **then** *class* =neolite
**else if** $f_{57} = 198$ **then** *class* =pacman
**else if** $f_{61} = 100$ **then** *class* =pecompact
**else if** $f_{60} = 30$ **then** *class* =telock
**else if** $f_{55} = 190$ **then** *class* =upx
**else if** $f_{32} = 0$ **and** $f_{53} = 186$ **then** *class* =stealth
**else if** $f_{26} = 0$ **and** $f_{48} = 96$ **then** *class* =aspack
**else if** $f_{33} = 1$ **and** $f_{54} = 233$ **then** *class* =aspack
**else if** $f_{17} = 0$ **and** $f_{116} = 10$ **then** *class* =armadillo
**else if** $f_{40} = 1$ **and** $f_{114} = 7$ **then** *class* =armadillo
**else if** $f_{112} = 3$ **and** $f_{102} = 139$ **then** *class* =armadillo
**else if** $f_{36} = 11$ **and** $f_{38} = 6$ **then** *class* =nspack

**else if** $f_{25} = 6$ **and** $f_{35} = 1$ **then** *class* =aspack
**else if** $f_{53} = 186$ **then** *class* =stealth
**else if** $f_{21} = 3$ **and** $f_{114} = 7$ **then** *class* =armadillo
**else if** $f_{14} = 13824$ **then** *class* =nspack
**else if** $f_{42} = 27$ **then** *class* =nspack
**else if** $f_{21} = 3$ **and** $f_{81} = 86$ **then** *class* =armadillo
**else** *class* =not packed

*Appendix C.4. Decision Trees*

**if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} <= 10.5$ **and** $f_{60} > 254.5$ **then** *class* =upx
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} > 237.5$ **then** *class* =enigma
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} > 232.5$ **then** *class* =neolite
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} > 10.5$ **then** *class* =armadillo
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} > 252.0$ **then** *class* =mpress
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} > 254.5$ **and** $f_{116} <= 0.163$ **then** *class* =vmprotect
**else if** $f_{40} > 0.276$ **and** $f_{52} > 254.5$ **then** *class* =telock
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} > 254.5$ **and** $f_{116} > 0.163$ **then** *class* =pecompact
**else if** $f_{40} <= 0.276$ **then** *class* =kkrunchy
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} > 254.5$ **then** *class* =pacman
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} <= 10.5$ **and** $f_{60} <= 254.5$ **and** $f_{47} > 7.605$ **then** *class* =stealth
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} <= 10.5$ **and** $f_{60} <= 254.5$ **and** $f_{47} <= 7.605$ **and** $f_{36} > 0.5$ **then** *class* =aspack
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} <= 10.5$ **and** $f_{60} <= 254.5$ **and** $f_{47} <= 7.605$ **and** $f_{36} <= 0.5$ **and** $f_{44} <= 7.899$ **then** *class* =not packed
**else if** $f_{40} > 0.276$ **and** $f_{52} <= 254.5$ **and** $f_{68} <= 252.0$ **and** $f_{93} <= 254.5$ **and** $f_{56} <= 254.5$ **and** $f_{49} <= 232.5$ **and** $f_{54} <= 237.5$ **and** $f_{115} <= 10.5$ **and** $f_{60} <= 254.5$ **and** $f_{47} <= 7.605$ **and** $f_{36} <= 0.5$ **and** $f_{44} > 7.899$ **then** *class* =nspack

*Appendix C.5. RIPPER*

**if** $f_{49} >= 103$ **and** $f_{49} < 110$ **then** *class* =vmprotect
**else if** $f_{16} >= 768$ **and** $f_{16} < 1.28e+03$ **and** $f_{37} >= 1.024$ **and** $f_{37} < 1.036$ **then** *class* =nspack
**else if** $f_{101} >= 245.5$ **then** *class* =nspack
**else if** $f_{9} >= 602$ **and** $f_{9} < 7.2e+04$ **and** $f_{81} >= 83.5$ **and** $f_{81} < 95.5$**then** *class* =nspack
**else if** $f_{87} >= 254.5$ **and** $f_{113} => 1.5$ **and** $f_{113} < 2.5$ **then** *class* =nspack
**else if** $f_{14} >= 6.01e+06$ **and** $f_{14} < 6.8e+06$ **then** *class* =kkrunchy
**else if** $f_{54} >= 78$ **and** $f_{54} < 82.5$ **then** *class* =pecompact
**else if** $f_{54} >= 182$ **and** $f_{54} < 190$ **then** *class* =stealth
**else if** $f_{25} < 0.5$ **and** $f_{110} >= 76.5$ **and** $f_{110} < 99.5$ **then** *class* =aspack
**else if** $f_{16} >= 768$ **and** $f_{16} < 1.28e+03$ **and** $f_{81} > 0.5$ **then** *class* =aspack
**else if** $f_{115} >= 10.5$ **and** $f_{115} < 15.5$ **then** *class* =armadillo
**else if** $f_{50} >= 187$ **and** $f_{50} < 191$ **then** *class* =upx

**else if** $f_{55} >= 86.5$ **and** $f_{55} < 89$ **then** *class* =mpress
**else if** $f_{57} >= 15.5$ **and** $f_{57} < 16.5$ **then** *class* =enigma
**else if** $f_{55} >= 89$ **and** $f_{55} < 92$ **then** *class* =pacman
**else if** $f_{50} >= 165.5$ **and** $f_{50} < 167$ **then** *class* =neolite
**else if** $f_{36} < 0.5$ **then** *class* =not packed
**else** *class* =ntelock

## References


[1] XGBoost. `https://xgboost.readthedocs.io/`. Accessed: 2022-04-15.

[2] Neda Abdelhamid and Fadi Thabtah. Associative classification approaches: review and comparison. *Journal of Information & Knowledge Management*, 13(03):1450027, 2014.

[3] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. Associative classification mining for website phishing classification. In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer ..., 2013.

[4] Hussein Abu-Mansour, Wa'el Hadi, TL McCluskey, and Fadi Thabtah. Associative text categorisation rules pruning method. In *1st International Symposium on Linguistic and Cognitive Approaches to Dialog Agents: A Symposium at the AISB 2010 Convention*, pages 39–44. The Society for the Study of Artifical Intelligence and the Simulation of ..., 2010.

[5] Maher Aburrous, M Alamgir Hossain, Keshav Dahal, and Fadi Thabtah. Intelligent phishing detection system for e-banking using fuzzy data mining. *Expert systems with applications*, 37(12): 7913–7921, 2010.

[6] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[7] Jaber Alwidian, Bassam H Hammo, and Nadim Obeid. Wcba: Weighted classification based on association rules algorithm for breast cancer disease. *Applied Soft Computing*, 62:536–549, 2018.

[8] Simon Andrews. A new method for inheriting canonicity test failures in Close-by-One type algorithms. In *The 14th International Conference on Concept Lattices and Their Applications*, 2018.

[9] John OR Aoga, Tias Guns, Siegfried Nijssen, and Pierre Schaus. Finding probabilistic rule lists using the minimum description length principle. In *International Conference on Discovery Science*, pages 66–82. Springer, 2018.

[10] Elena Baralis, Silvia Chiusano, and Paolo Garza. A lazy approach to associative classification. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):156–171, 2007.

[11] Munkhbayar Bat-Erdene, Taebeom Kim, Hyundo Park, and Heejo Lee. Packer detection for multi-layer executables using entropy analysis. *Entropy*, 19(3):125, 2017. doi: 10.3390/e19030125. URL https://doi.org/10.3390/e19030125.

[12] Munkhbayar Bat-Erdene, Hyundo Park, Hongzhe Li, Heejo Lee, and Mahn-Soo Choi. Entropy analysis to classify unknown packing algorithms for malware detection. *International Journal of Information Security*, 16(3):227–248, 2017.

[13] Fabrizio Biondi, Michael A. Enescu, Thomas Given-Wilson, Axel Legay, Lamine Noureddine, and Vivek Verma. Effective, efficient, and robust packing detection and classification. *Computers & Security*, 85:436–451, 2019. doi: 10.1016/j.cose.2019.05.007. URL https://doi.org/10.1016/j.cose.2019.05.007.

[14] Jadzia Cendrowska. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349 – 370, 1987. ISSN 0020-7373. doi: https://doi.org/10.1016/S0020-7373(87)80003-2. URL http://www.sciencedirect.com/science/article/pii/S0020737387800032.

[15] Fuzan Chen, Yanlan Wang, Minqiang Li, Harris Wu, and Jin Tian. Principal association mining: an efficient classification approach. *Knowledge-Based Systems*, 67:16–25, 2014.

[16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[17] Yang-Seo Choi, Ik kyun Kim, Jin-Tae Oh, and Jae cheol Ryou. PE File Header Analysis-Based Packed PE File Detection Technique (PHAD). *International Symposium on Computer Science and its Applications*, pages 28–31, 2008.

[18] William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[19] Wa'el Hadi, Faisal Aburub, and Samer Alhawari. A new fast associative classification algorithm for detecting phishing websites. *Applied Soft Computing*, 48:729–734, 2016.

[20] Michael Hahsler, Ian Johnson, Tomáš Kliegr, and Jaroslav Kuchař. Associative Classification in R: arc, arulesCBA, and rCBA. *The R Journal*, 11(2):254–267, 2019. doi: 10.32614/RJ-2019-048. URL https://doi.org/10.32614/RJ-2019-048.

[21] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

[22] Rui Henriques and Sara C Madeira. Flebic: Learning classifiers from high-dimensional biomedical data using discriminative biclusters with non-constant patterns. *Pattern Recognition*, page 107900, 2021.

[23] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[24] Neminath Hubballi and Himanshu Dogra. Detecting packed executable file: Supervised or anomaly detection method? *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 638–643, 2016.

[25] Radek Janostik, Jan Konecny, and Petr Krajča. Lcm from fca point of view: A cbo-style algorithm with speed-up features. *International Journal of Approximate Reasoning*, 142:64–80, 2022.

[26] Guhyeon Jeong, Euijin Choo, Joosuk Lee, Munkhbayar Bat-Erdene, and Heejo Lee. Generic unpacking using entropy analysis. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 98–105. IEEE, 2010.

[27] Ruoming Jin and Gagan Agrawal. Frequent pattern mining in data streams. In *Data Streams*, pages 61–84. Springer, 2007.

[28] Kesav Kancherla, J Kevin Donahue, and Srinivas Mukkamala. Packer identification using byte plot and Markov plot. *Journal of Computer Virology and Hacking Techniques*, 12:101–111, 2015.

[29] Jan Konecny and Petr Krajča. Systematic categorization and evaluation of cbo-based algorithms in fca. *Information Sciences*, 575:265–288, 2021.

[30] Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Proceedings 2001 IEEE international conference on data mining*, pages 369–376. IEEE, 2001.

[31] Xueming Li, Dongxia Qin, and Cun Yu. Accf: Associative classification based on closed frequent itemsets. In *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, volume 2, pages 380–384. IEEE, 2008.

[32] Chun-Wei Lin, Tzung-Pei Hong, and Wen-Hsiang Lu. The pre-fufp algorithm for incremental mining. *Expert Systems with Applications*, 36(5):9498–9505, 2009.

[33] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining*, 1998.

[34] Bing Liu, Yiming Ma, and Ching-Kian Wong. Classification using association rules: weaknesses and enhancements. In *Data mining for scientific and engineering applications*, pages 591–605. Springer, 2001.

[35] José María Luna, Philippe Fournier-Viger, and Sebastián Ventura. Frequent itemset mining: A 25 years review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(6):e1329, 2019.

[36] Robert Lyda and James Hamrock. Using entropy analysis to find encrypted and packed malware. *IEEE Security & Privacy*, 5(2):40–45, 2007. doi: 10.1109/MSP.2007.48. URL `https://doi.org/10.1109/MSP.2007.48`.

[37] Tatiana Makhalova, Sergei O Kuznetsov, and Amedeo Napoli. Closure structure: a deeper insight. *FCA4AI 2020*, page 45, 2020.

[38] Malwarebytes Labs. 2020 state of malware report, 2020. URL `https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf`.

[39] MicroSoft. Microsoft digital defense report, september 2020, 2020. URL `https://download.microsoft.com/download/f/8/1/f816b8b6-bee3-41e5-b6cc-e925a5688f61/Microsoft_Digital_Defense_Report_2020_September.pdf`.

[40] Francisco Padillo, Jose Maria Luna, and Sebastian Ventura. Lac: Library for associative classification. *Knowledge-Based Systems*, 193:105432, 2020.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of packed executables for accurate computer virus detection. *Pattern Recognition Letters*, 29(14):1941 – 1946, 2008. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2008.06.016. URL `http://www.sciencedirect.com/science/article/pii/S0167865508002110`.

[43] Viet Phan-Luong and Rabah Messouci. Building classifiers with association rules based on small key itemsets. In *2007 2nd International Conference on Digital Information Management*, volume 1, pages 200–205. IEEE, 2007.

[44] Jithu Raphel and P. Vinod. Information theoretic method for classification of packed and encoded files. In *Proceedings of the 8th International Conference on Security of Information and Networks*, SIN '15, pages 296–303, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3453-2. doi: 10.1145/2799979.2800015. URL `http://doi.acm.org/10.1145/2799979.2800015`.

[45] Zhen-Hui Song and Yi Li. Associative classification over data streams. In *2010 2nd International Conference on Information Engineering and Computer Science*, pages 1–4. IEEE, 2010.

[46] Li Sun, Steven Versteeg, Serdar Boztaş, and Trevor Yann. Pattern recognition techniques for the classification of malware packers. In *Proceedings of the 15th Australasian Conference on Information Security and Privacy*, ACISP'10, pages 370–390, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-14080-7, 978-3-642-14080-8. URL `http://dl.acm.org/citation.cfm?id=1926211.1926239`.

[47] Fadi Thabtah, Peter Cowling, and Yonghong Peng. Mcar: multi-class classification based on association rule. In *The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005.*, page 33. IEEE, 2005.

[48] Fadi Abdeljaber Thabtah. A review of associative classification mining. *Knowledge Engineering Review*, 22(1):37–65, 2007.

[49] The Shadowserver Foundation. The shadowserver foundation, 2020. URL `https://www.shadowserver.org/`. Library Catalog: www.shadowserver.org.

[50] Xabier Ugarte-Pedrero, Igor Santos, and Pablo G. Bringas. Structural feature based anomaly detection for packed executable identification. In Álvaro Herrero and Emilio Corchado, editors, *Computational Intelligence in Security for Information Systems*, pages 230–237, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21323-6.

[51] Xabier Ugarte-Pedrero, Igor Santos, Iván García-Ferreira, Sergio Huerta, Borja Sanz, and Pablo G Bringas. On the adoption of anomaly detection for packed executable filtering. *Computers & Security*, 43:126–144, 2014.

[52] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 77–86, 2005.

[53] Sebastián Ventura and José María Luna. *Supervised descriptive pattern mining*. Springer, 2018.

[54] Rosana Veroneze and Fernando J Von Zuben. Scalability achievements for enumerative biclustering with online partitioning: case studies involving mixed-attribute datasets. *Engineering Applications of Artificial Intelligence*, 2021. doi: https://doi.org/10.1016/j.engappai.2020.104147.

[55] Rosana Veroneze, Arindam Banerjee, and Fernando J Von Zuben. Enumerating all maximal biclusters in numerical datasets. *Information Sciences*, 379:288–309, 2017.

[56] Yanfang Ye, Qingshan Jiang, and Weiwei Zhuang. Associative classification and post-processing techniques used for malware detection. In *2008 2nd International Conference on Anti-counterfeiting, Security and Identification*, pages 276–279. IEEE, 2008.

[57] Mohaddeseh Zakeri, Fatemeh Faraji Daneshgar, and Maghsoud Abbaspour. A static heuristic approach to detecting malware targets. *Security and Communication Networks*, 8(17):3015–3027, 2015. doi: 10.1002/sec.1228. URL `https://doi.org/10.1002/sec.1228`.

[58] Mohammed J Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002.

**Khanh Huu The Dam** is a postdoctoral researcher at the the Louvain School of Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium. He received his PhD from the Paris 7 University in 2018. He is interested in program analysis, malware detection, machine learning and data mining.

**Thomas Given-Wilson** is a research associate at Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium. He received his PhD in computer science from the University of Technology, Sydney, Australia. He has worked for National Information and Communication Technology Australia on program analysis and formal methods. Then moved to Institut national de recherche en informatique et en automatique in Saclay and later Rennes (both France) working on programming languages, concurrency, formal methods, machine learning, privacy, security, encryption, and program & malware analysis. At UCL he works on machine learning, formal methods, security, program analysis, program repair, privacy, healthcare, and software security and robustness and primary investigator for two projects.

**Axel Legay** is a professor at Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium where he leads the security team. He has held positions at University of Liège and CMU (under the supervision of Ed Clarke). He led the Threat Assessment and Mitigation for Information Security (TAMIS) team at Institut national de recherche en informatique et en automatique Rennes. His main research interests are in developing formal specification and verification techniques for Software Engineering and security including malware and program analysis. He referees for top journals and

conferences in formal verification and simulation, and program co-chair of INFINITY'09, FIT'10, Runtime Verification 2013, Splat 2014, FORMATS 2014, ATVA 2016, and TACAS 2017. He has been the primary investigator for more than 15 projects over the eight last years.

**Rosana Veroneze** is a postdoctoral researcher at the Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, University of Campinas (Unicamp), Brazil. She is currently also a visiting researcher at Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium. She received her PhD in computer engineering from Unicamp in 2016. Her research interests include artificial intelligence, data mining and machine learning areas.